Universidad Carlos III de Madrid

GRADO EN INGENIERÍA DE SISTEMAS DE COMUNICACIONES

IMPLEMENTACIÓN DE UN SISTEMA PARA GESTIÓN DE VERSIONES SOFTWARE BASADO EN INCIDENCIAS

Autor: Rafael Roldán Tormo

Tutora: Dra. Florina Almenares Mendoza (Universidad Carlos III de Madrid)

Directora: Yajaira Salgado Lorenzo (INDRA Sistemas S.A.)

Leganés, Enero de 2014

Título: Implementación de un Sistema para Gestión de Versiones Software basado en Incidencias.

Autor: Rafael Roldán Tormo

Tutora: Florina Almenares Mendoza

EL TRIBUNAL

Presidente:_____

Vocal:_____

Secretario:_____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de ____ de 20___ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

AGRADECIMIENTOS

En primer lugar, me gustaría dar las gracias a todo el departamento SCV de INDRA, por la oportunidad que me han dado de poder realizar el proyecto fin de carrera con ellos. Gracias a esta oportunidad, he aprendido mucho dentro del mundo laboral y me servirá en un futuro próximo. Pero sobre todo, quiero dar las gracias a mis compañeros Guillermo, Hugo y José Miguel. Gracias por cada segundo que habéis aportado de vuestro valioso tiempo a mi proyecto, sin vosotros, este proyecto no habría salido adelante.

Gracias a Florina y Yajaira, por aconsejarme y ayudarme cada vez que os he necesitado para algo. Gracias por aportar vuestro granito de arena a este proyecto.

También me gustaría agradecer a todas las maravillosas personas que he conocido durante estos años desde que salí de Cabra, mi ciudad natal, y que gracias a ellas, hoy he llegado al final de esta etapa: Leganés, Madrid, Sevilla, Cádiz, Boston, Chicago, Canadá, Inglaterra, Ecuador o México.

Gracias a mis amigos egabrenses, por estar ahí a pesar de la distancia. Gracias por todos estos años y los que quedan… Carlos, Pino, Paco Rosa, Paco Málaga, Pavón, Salva, Javi, Fabi, Víctor, Muñoz y Ropero.

El fin de esta etapa no hubiera sido posible sin ellos. Gracias papá, gracias mamá y gracias hermana por apoyarme en todas y cada una de mis decisiones. Gracias por darme siempre ese empujoncito que necesitaba en los momentos difíciles. Gracias por todo.

RESUMEN

La eficiencia es uno de los factores más determinantes en la actualidad para que las empresas de desarrollo tecnológico consigan cumplir los plazos establecidos de entrega de proyectos.

El desarrollo de tecnología software y hardware en grandes empresas como INDRA SISTEMAS, requiere un control minucioso de los sistemas, almacenando y gestionando toda la información de los mismos. El control de toda esta información ayuda en gran medida a que los desarrollos sean más eficaces y eficientes. El SCV (Sistema de Comunicaciones de Voz Digital), es uno de los departamentos de INDRA encargados en el desarrollo de nuevas tecnologías software y hardware para el control de tráfico aéreo internacional.

Este proyecto propone el diseño de una aplicación que permita la gestión de toda la información que se encuentra entre todos los proyectos software para los que trabaja el departamento SCV (información sobre el versionado de cada proyecto software y las incidencias software / hardware que han llevado a una nueva versión del proyecto). El desarrollo de esta nueva herramienta de gestión, permitirá al departamento sustituir un archivo Excel donde almacenaban la información relativa a todos los proyectos, por una base de datos con interfaz gráfica, aumentando la seguridad de la información tratada y sobre todo y más importante, se aumentará la eficiencia gestionando la información.

Palabras clave: ficheros csv, incidencias, sistemas de comunicaciones, importación.

ABSTRACT

Nowadays, efficiency is one of the most determining factors for the technology development enterprises to meet the delivery deadlines for projects.

The software and hardware technology development of big enterprises such as INDRA SISTEMAS requires a detailed control of the systems, storing and managing all its information. Controlling all this information greatly helps developments to be more efficient and productive. The SCV (Digital Voice Communications System) is one of the INDRA's departments in charge of new software and hardware technology for international air traffic control.

This project proposes the design of an application that allows to process all the information located within the totality of SCV department software projects (information about the versioning of the software projects and the software / hardware incidences due to there is a new version in a project). The development of this management tool will let the department to substitute the excel file where all the projects information were stored by a database with a graphic interface, increasing its security and, what is even more important, its efficiency processing the information.

Keywords: csv files, incidences, communications systems, importation.

Tabla de contenido

Índice Tablas

Índice Figuras

1. INTRODUCCIÓN

La eficiencia es uno de los factores más determinantes en la actualidad para que las empresas de desarrollo tecnológico consigan cumplir los plazos establecidos de entrega de proyectos.

El objetivo del presente documento es la de describir el proyecto fin de grado "Implementación de un sistema para gestión de versiones software basado en incidencias". Este proyecto consiste en el análisis, diseño e implementación de una aplicación que permita el almacenamiento y gestión de la información software que puede haber en cualquier sistema.

A continuación, se presentarán las motivaciones que han dado lugar a la realización de este proyecto fin de grado, los objetivos a alcanzar, los acrónimos y los términos empleados.

1.1 MOTIVACIÓN

INDRA es una empresa multinacional española dedicada a la consultoría y tecnologías de la información. Es una de las más importantes del sector a nivel mundial. Uno de los departamentos de INDRA es el SCV (Sistema de Comunicaciones de Voz Digital).

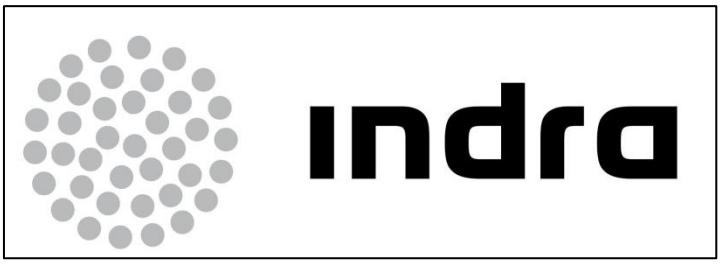


Figura 1: Logotipo de INDRA

El departamento SCV se encarga de la integración de voz y datos gracias a la tecnología proporcionada por la plataforma hardware y software de la empresa. Se consigue la más alta flexibilidad y efectividad para todo tipo de soluciones dentro de un entorno de control de tráfico aéreo. Es decir, este departamento se encarga de diseñar, crear y mejorar los sistemas de comunicaciones de la mayoría de los aeropuertos internacionales (España, Ecuador, Argentina, India, Pakistán, Australia, Colombia, Omán, Bosnia, etc.).

Estos sistemas de comunicaciones funcionan igual que otros sistemas que a lo mejor conocemos más en profundidad y con los que trabajamos todos los días. El sistema operativo móvil de Apple, IOS, es un sistema que está continuamente desarrollándose y mejorándose, es decir, cada vez que sale al mercado una actualización del sistema operativo IOS, implica una serie de mejoras y correcciones de errores detectadas en versiones anteriores.

Los sistemas de comunicaciones de aeropuertos que desarrolla el departamento SCV tienen la misma rutina. INDRA entrega una versión del sistema al aeropuerto correspondiente. A esa versión, se le encontrarán incidencias software o hardware (Ptrs) que habrá que solucionar, por lo que el departamento en un futuro entregará una nueva versión del sistema solucionando estas incidencias y mejorando otras funciones del sistema. Se puede observar el proceso en el siguiente diagrama:

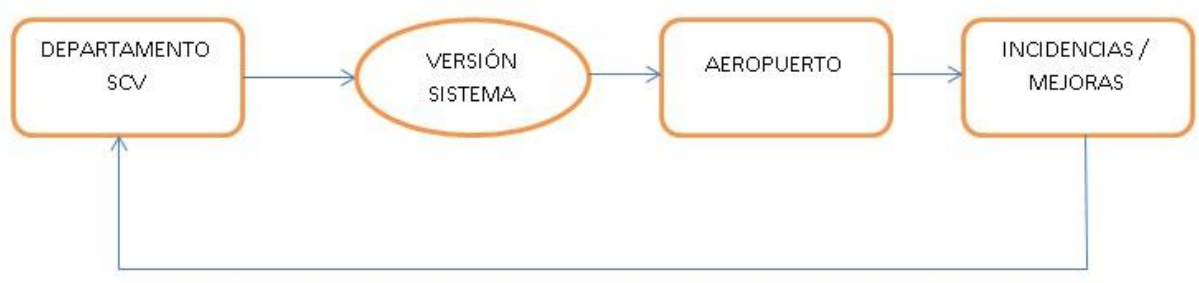


Figura 2: Proceso Sistema de Comunicación

El sistema de comunicación internamente está formado por una gran cantidad de componentes software. Cada uno de estos componentes tiene su propia versión software, que será modificada o no, en función de si la versión del sistema que sea lanzada ha tenido que tocar dicho componente para alguna mejora o para solucionar una incidencia.

Todo este cambio de versiones (tanto versiones del sistema como versiones de componentes software) afecta a cada uno de los aeropuertos para los cuales trabaja el departamento SCV, por lo que se necesita tener controlado todo el versionado software, para saber en todo momento que hay instalado en cada aeropuerto, además de tener controlado el historial de versiones.

Actualmente, el departamento SCV lleva un control de los sistemas de comunicación poco seguro. El versionado de los sistemas se encuentra almacenado en un fichero Excel. Es un sistema poco seguro debido a que se pueden introducir errores a la hora de actualizar el fichero o es muy poco eficiente cuando se quiere buscar el historial software de una versión concreta de un proyecto.

El presente proyecto fin de grado tiene como finalidad la de solucionar este control de versiones que el departamento SCV tiene en la actualidad, creando una aplicación que de forma eficiente permita gestionar toda esta información.

1.2 OBJETIVOS

El principal objetivo del presente proyecto fin de grado **es implementar un sistema para gestionar el versionado software de los sistemas de comunicaciones de voz.** Este sistema tiene que disponer de las siguientes características:

- **El sistema final diseñado tiene que utilizar en su totalidad software libre.**
  La aplicación será utilizada en el departamento SCV de INDRA, por lo que, para evitar costes adicionales, se ha optado por el empleo de software libre.

- **La aplicación debe de ser sencilla y comprensible.**
  Cuando un usuario del departamento SCV decida utilizar la aplicación, debe de ser capaz de utilizar la herramienta sin ninguna dificultad.

- **Utilización de ficheros con extensión csv.**
  La aplicación funcionará mediante ficheros csv (*Comma-Separated Values*). Cuando el equipo de desarrollo tiene lista una versión para ser lanzada a cualquiera de los aeropuertos para los que trabaja INDRA, genera unos ficheros csv con toda la información relativa a la versión. Estos ficheros son los que tienen que ser cargados en la aplicación a desarrollar para que sean almacenados en el sistema.

- **La información tiene que poder ser modificable.**
  Si la información almacenada en el sistema relativa a una versión es incorrecta, la aplicación permitirá corregir dicha información cargando el fichero csv correcto de la versión afectada.

- **Gestión / Visualización rápida de la información.**
  La aplicación permitirá una búsqueda rápida y eficaz de la información relativa a una versión del sistema. Esto será una de las principales mejoras y comodidades respecto al control de versionado que el departamento tiene en la actualidad, el ya mencionado fichero Excel.

1.3 MEDIOS EMPLEADOS

Para la realización del presente proyecto se han empleado una serie de herramientas hardware y software.

El hardware utilizado es el siguiente:

- Ordenador portátil Sony Vaio, serie VGN-NS11S.
- Ordenador portátil Toshiba, Satellite Pro S200.

El software empleado es el siguiente:

- Microsoft Windows 8.
- Eclipse, versión Kepler.
- PgAdmin III.
- Microsoft Visual Studio 2008.
- QT Designer.
- Microsoft Office Hogar y Estudiantes 2010.
- Microsoft Visio

1.4 TERMINOLOGÍA

En el siguiente apartado del capítulo, se exponen una serie de acrónimos, abreviaciones y definiciones claves para poder entender el contenido completo de la memoria.

1.4.1 Acrónimos y abreviaturas

ASD: *Adaptative Software Development*.

ATM: *Air Traffic Management*.

CSCI: *Computer Software Configuration Item*.

CSS: *Cascading Style Sheets*.

CSV: *Comma-Separated Values*.

CWP: *Controller's Working Position*.

DB: *Data Base*.

DDL: *Data Definition Language*.

FK: *Foreign Key*.

GUI: *Graphical User Interface*.

GWP: *Pasarela Gateway*.

H-CWP: *Hardware - Controller's Working Position*.

H-GWP: *Hardware – Pasarela Gateway*.

HMI: *Human Machine Interface*.

HPTR: *Hardware Problem Trouble Report*.

H-SeeSCV: Hardware - Sistema de Explotación de Estadísticas SCV.

H-TMCS: *Hardware – Management Position*.

HWCI: *Hardware Configuration Item*.

IOS: *Internet Operating System*.

LED: *Light-Emitting Diode*.

OLE: Incrustación y Enlazado de Objetos.

PK: *Primary Key*.

PTR: *Problem Trouble Report*.

RAE: Real Academia Española.

SCV: Sistema de Comunicación de Voz Digital.

SeeSCV: Sistema de Explotación de Estadísticas SCV.

SGBD: Sistema de Gestión de Bases de Datos.

SPTR: *Software Problem Trouble Report*.

SQL: *Structured Query Language*.

TMCS: *Management System*.

VAIO: *Video and Audio Integrated Operation*.

1.4.2 Definiciones

BD SCV: base de datos que se encuentra dentro de la herramienta IBM Rational Change, en la cual, el cliente reporta las incidencias encontradas en su sistema.

BD SCV_INT: base de datos que se encuentra dentro de la herramienta IBM Rational Change, en la cual, el equipo de pruebas del departamento SCV reportan las incidencias detectadas en los diferentes sistemas.

COTS: se denomina COTS, aplicándolo a INDRA, al conjunto de software y hardware que no es fabricado por la empresa, pero que luego es vendido como parte de sus productos.

HPTR Concluded: incidencia hardware que ha sido solucionada y se encuentra incluida en la siguiente versión del sistema.

HPTR in acceptance test: incidencia hardware que ha sido solucionada y se encuentra incluida en la siguiente versión del sistema. Se diferencia de la HPTR Concluded en que esta incidencia fue detectada por el cliente, por lo que la solución tiene que ser validada por el mismo.

IBM Rational Change: herramienta utilizada en el departamento SCV de INDRA para gestionar las incidencias de todos los proyectos para los que trabaja el departamento.

SPTR Concluded: incidencia software que ha sido solucionada y se encuentra incluida en la siguiente versión del sistema.

SPTR in acceptance test: incidencia software que ha sido solucionada y se encuentra incluida en la siguiente versión del sistema. Se diferencia de la SPTR Concluded en que esta incidencia fue detectada por el cliente, por lo que la solución tiene que ser validada por el mismo.

Widget: es una pequeña aplicación o programa que tiene entre sus objetivos dar fácil acceso a funciones frecuentemente usadas y proporcionar información visual. Los widgets se suelen utilizar para ser insertados en otras páginas webs o aplicaciones.

1.5 ESTRUCTURA DE LA MEMORIA

A continuación, se incluye un breve resumen de cada uno de los capítulos por los que está estructurada la memoria.

En el **primer capítulo**, "Introducción", se han introducido los motivos que han llevado a la elección de este proyecto fin de carrera, además de los objetivos que se plantean conseguir con el desarrollo del mismo. También se ha incluido una serie de acrónimos, abreviaturas y definiciones para facilitar la lectura y comprensión de la memoria.

En el **segundo capítulo**, "Planteamiento del problema", se analizan las tecnologías que están relacionadas con el presente proyecto y se seleccionan las que se van a emplear para llevar a cabo el desarrollo, argumentando los motivos de la elección. Esto se encuentra incluido dentro del "Estado del arte". Por último, se recogen las reglas y normas que afectan al proyecto que se está realizando.

En el **tercer capítulo**, "Análisis y diseño de la solución técnica", se realiza un estudio de las metodologías de desarrollo software más utilizadas en la actualidad, y se determina la metodología a emplear en el proyecto, argumentando la elección. También se lleva a cabo una descripción detallada sobre el resultado esperado del sistema. Se explican los casos de uso de la aplicación a desarrollar y por último, se expone de manera clara y precisa los requisitos y funcionalidades que tendrá el sistema.

En el **cuarto capítulo**, "Diseño", se explicará de manera detallada el diseño de la aplicación desarrollada y las partes de las que se compone.

En el **quinto capítulo**, "Plan de pruebas", se detallará el conjunto de pruebas que se realizarán al sistema para verificar que se han cumplido los objetivos y requisitos del proyecto.

El **sexto capítulo**, "Gestión del proyecto", recoge la planificación temporal que se ha seguido para el desarrollo de la aplicación. También se detalla el presupuesto total necesario para realizar el proyecto.

En el **séptimo y último capítulo**, "Conclusiones y líneas futuras", se recogen las conclusiones sacadas tras la finalización del proyecto fin de carrera. Además, se incluyen una serie de mejoras que se realizarán para mejorar la aplicación en un futuro.

2. PLANTEAMIENTO DEL PROBLEMA

Para el desarrollo de un proyecto basado en una aplicación, es necesario analizar el contexto actual, las tecnologías que involucre y las aplicaciones que actualmente se encuentran en el mercado y que estén relacionadas con nuestro desarrollo. Con esto, lo que se pretende es dejar constancia de las ventajas y desventajas de nuestra aplicación, así como las tecnologías empleadas para el desarrollo de la aplicación.

En este capítulo, se expondrá la información obtenida y que posteriormente será tratada a lo largo del desarrollo del proyecto. En primer lugar, se llevará a cabo un estudio sobre bases de datos, analizando los distintos tipos de Sistemas de gestión de bases de datos, con el objetivo de determinar la que mejor se adapta a las necesidades de la aplicación a desarrollar. Posteriormente, se hará otro análisis sobre distintos tipos de lenguajes de programación, con el objetivo de determinar aquel que se adapte mejor a los requisitos de interfaz gráfica de usuario. Seguido de esto, se hará un estudio sobre otros sistemas de gestión de versiones que actualmente se encuentran en el mercado, con una argumentación de las ventajas que presenta la aplicación a desarrollar con respecto a estos sistemas. Para terminar, se recogen las reglas y normas que afectan al proyecto que se va a realizar.

2.1 ANÁLISIS DEL ESTADO DEL ARTE

2.1.1 Bases de datos

Una base de datos [1] es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente de forma organizada y estructurada.

El almacenamiento, modificación y extracción de la información de la base de datos se realiza gracias al Sistema de Gestión de Base de Datos (SGBD). Los datos de una base de datos, sólo pueden ser removidos de la misma por una solicitud explicita al Sistema Gestor de Base de Datos, por lo que podemos definir una base de datos como un conjunto de datos persistentes que es utilizado por los sistemas de aplicación.

Los SGBD también proporcionan métodos para mantener la integridad de los datos, para administrar el acceso de usuarios a los datos y para recuperar la información si el sistema se corrompe. En resumen, el principal objetivo de los SGBD es la organización de las base de datos y por lo tanto, hacer rápidas escrituras y lecturas de datos.

Las bases de datos han estado en uso desde los primeros días de los ordenadores electrónicos. A diferencia de los sistemas modernos, que se pueden aplicar a datos y necesidades muy diferentes, la mayor parte de los sistemas originales estaban enfocados a bases de datos específicas y, pensados para ganar velocidad a costa de perder flexibilidad. Originalmente, los SGBD solo estaban disponibles para las grandes organizaciones que podían disponer de los complejos ordenadores.

El Sistema de Base de Datos se define como el conjunto de base de datos y Sistema de Gestión de Base de Datos (SGBD).

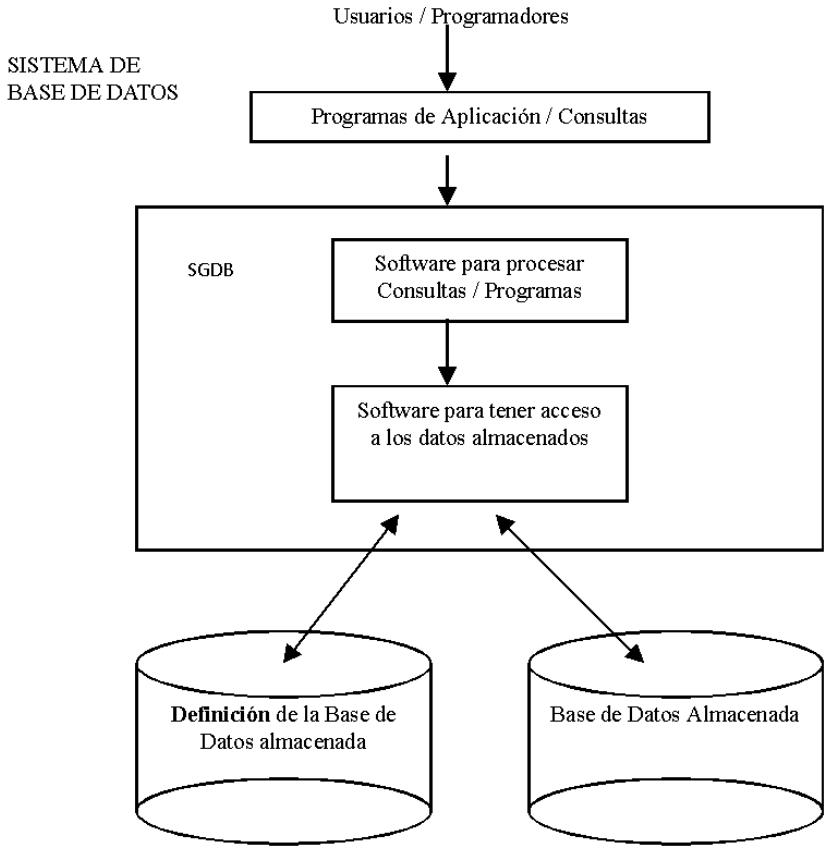


Figura 3: Sistema de Base de Datos

2.1.1.1 Sistemas de gestión de bases de datos relacionales

Se define un modelo de datos como una colección de conceptos que se emplean para describir la estructura de una base de datos. Esa colección de conceptos incluye entidades, atributos y relaciones.

Los modelos de datos se clasifican en:

- Modelos conceptuales, facilitan la descripción global del conjunto de información al nivel más próximo al usuario, acercándose al mundo real.
- Los modelos lógicos, están orientados a describir los datos a nivel lógico para el SGBD.

Un SGBD relacional [2], es aquel que cumple el modelo relacional. Este modelo es el más utilizado en la actualidad para implementar bases de datos. Permiten establecer interconexiones entre los datos que se encuentran guardados en tablas, y mediante dichas conexiones relacionar los datos de ambas tablas.

Las características más importantes de una Base de Datos relacional son:

- Se compone de varias tablas o relaciones.
- No pueden existir dos o más tablas que contengan el mismo nombre o registro.
- Cada tabla es a su vez un conjunto de filas y columnas (registros).
- Las claves primarias (PK, *primary key*) son la clave principal de un registro dentro de una tabla y deben cumplir con la integridad de datos.
- Las claves ajenas (FK, *foreign key*) se colocan en la tabla hija y contienen el mismo valor que la clave primaria del registro padre. Gracias a estas claves se hacen las relaciones.

Actualmente, los SGBD más comunes son: MySQL, Oracle, PostgreSQL y SQL Server. Todos ellos utilizan el lenguaje de consulta SQL, que es un lenguaje declarativo de acceso a base de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

Figura 4: Tablas con modelo relacional

2.1.1.2 Normalización de bases de datos

El proceso de normalización de una base de datos consiste en aplicar una serie de reglas con el objetivo de:

- Evitar que haya redundancia en los datos.
- Evitar problemas a la hora de actualizar los datos de las tablas.
- Proteger la integridad de los datos.

Existen 3 niveles de normalización que deben respetarse para poder decir que nuestra base de datos está normalizada:

1. **La Primera Forma Normal:** para respetar esta primera forma, no se pueden repetir datos en las tablas de nuestra base de datos.
2. **La Segunda Forma Normal:** si se ha respetado previamente la primera forma, esta segunda forma para respetarse, cada columna de la tabla debe depender de la clase. Esto quiere decir que, todo registro debe depender únicamente de la clave principal, y en el caso de que tuviéramos alguna columna que se repitiera a lo largo de todos los registros, estos datos deberían de pasarse a una nueva tabla.

3. **La Tercera Forma Normal:** la tercera forma normal habla de que ninguna columna puede estar dependiendo de una columna que no tenga clave, y de que no pueden existir datos derivados.

2.1.1.3 Microsoft SQL Server

Producido por Microsoft [9], es un sistema de gestión de base de datos que se basa en el modelo relacional. La principal función de Microsoft SQL Server es la de almacenar y consultar los datos solicitados por otras aplicaciones que no tienen que estar obligatoriamente en el mismo ordenador, si se encuentran conectadas a través de una red local o a través de internet.

Las características fundamentales de este gestor de base de datos y que hacen que sea uno de los más comunes en la actualidad son las siguientes:

1. Seguridad.
2. Integridad de datos.
3. Concurrencia.
4. Recuperación.
5. Diccionario de datos.

Lo más importante de Microsoft SQL Server es el almacenamiento y servicios en la nube, además de que los datos se mantienen siempre organizados y accesibles.

2.1.1.4 Oracle

Desarrollado por Oracle Corporation [10], es un sistema de gestión de base de datos que se basa en el modelo relacional.

Las características principales de Oracle son:

1. Gestión de grandes bases de datos.
2. Usuarios concurrentes.
3. Alto rendimiento en transacciones.
4. Compatibilidad.
5. Gestión de la seguridad.
6. Contestabilidad.
7. Sistema de alta disponibilidad.

Es el motor de base de datos relacional más usado a nivel mundial y puede ser ejecutado en todas las plataformas. Otra de las ventajas de Oracle es que permite el uso de particiones para mejorar la eficiencia.

Oracle es el referente en sistemas de gestores de base de datos.

2.1.1.5 PostgreSQL

Es un sistema gestor de base de datos relacional orientado a objetos y libre. Al igual que otros muchos proyectos de código abierto, el desarrollo de PostgreSQL [4] se encuentra dirigido por una comunidad de desarrolladores que trabajan de forma libre y desinteresada (PostgreSQL Global Development Group).

Algunas de sus principales características son, entre otras [3]:

1. Triggers.
2. Vistas.
3. Consultas complejas.
4. Claves externas.
5. Alta concurrencia.
6. Integridad de las transacciones.

PostgreSQL es uno de los gestores de base de datos de código abierto más potente que se encuentra actualmente en el mercado.

2.1.1.6 MySQL

Es un sistema gestor de base de datos relacional propiedad de Oracle Corporation. Es rápido, robusto y fácil de usar. También se adapta bien a la administración de datos en un entorno de red.

MySQL [11] es ofrecido con licencia pública (licencia GNU GPL) que permite utilizar el programa, además de consultar y manipular el código fuente. Los usuarios que deseen utilizarlo en productos más privativos pueden contratar una licencia que se lo permita.

Las principales características de MySQL son:

1. Conectividad segura.
2. Transacciones y claves foráneas.
3. Subconjunto amplio del lenguaje SQL.
4. Está disponible en gran cantidad de sistemas y plataformas.
5. Replicación.

MySQL es uno de los gestores de bases de datos más importantes de la actualidad. La fiabilidad y la velocidad han conseguido que se convierta en una alternativa muy fuerte a los sistemas de bases de datos propietarias.

2.1.1.7 Comparativa

A continuación, se va a realizar una comparativa de los gestores de bases de datos mencionados anteriormente, mostrando las ventajas y desventajas de cada uno de ellos:

	SQL Server	Oracle	PostgreSQL	MySQL
Tipos de licencia:				
GPL	No	No	Sí	Sí
Comercial	Sí	Sí	No	Sí
API's / Conectores:				
C	Sí	Sí	Sí	Sí
C++	Sí	No	Sí	Sí
Java	Sí	Sí	Sí	Sí
ODBC	Sí	Sí	Sí	Sí
JDBC	Sí	Sí	Sí	Sí
Sistemas operativos destacados	Windows.	Windows, Unix y Linux.	Windows, Unix y Linux.	Windows, Unix, Linux, Mac OS X, Solaris, FreeBSD, OpenBSD.
Integridad de datos	Sí	Sí	Sí	Sí
Replicación	Sí	Sí	Sí	Sí
Escalabilidad / Confiabilidad	Muy larga / Alta	Muy larga / Muy alta	Muy larga / Alta	Muy larga / Muy alta

Tabla 1: Comparativa de Gestores de bases de datos

2.1.2 Lenguajes de programación

Según la Real Academia Española (RAE), se define lenguaje en el ámbito de la informática y de las telecomunicaciones, al conjunto de signos y reglas que permite la comunicación con un ordenador.

Un lenguaje de programación [12] es aquella estructura que envía ciertas instrucciones a un programa de ordenador. Dicha estructura posee una base sintáctica y semántica.

Existen dos tipos de lenguajes de programación claramente diferenciados:

- **Lenguajes de bajo nivel:** este tipo de lenguaje depende totalmente de la máquina, es decir, un programa que sea realizado con este tipo de lenguaje no puede ser utilizado en otra máquina distinta.
- **Lenguajes de alto nivel:** este tipo de lenguaje se encuentra más cercano al lenguaje natural que al lenguaje máquina. Solucionan problemas mediante el empleo de Estructuras Dinámicas de Datos. Un programa que se encuentre escrito en alto nivel, puede ser utilizado en cualquier máquina sin ningún problema.

El índice TIBOE recoge el ranking de los lenguajes de programación más usados en función de los ingenieros informáticos cualificados del todo el mundo que utilizan estos lenguajes, en función de cursos y proveedores de terceros. Según el índice de TIBOE de Enero de 2013, los lenguajes de programación más usados son: Java, C y C++.

2.1.2.1 Java

Creado por Sun Microsystems, Inc., Java [14] es un lenguaje de programación orientado a objetos que permite la creación de programas que pueden ser utilizados en cualquier ordenador y en cualquier sistema operativo. Con Java, los programas creados se denominan applets.

Las características principales de Java son:

1. Es un lenguaje simple.
2. Orientado a objetos.
3. Robusto.
4. Portable.
5. Dinámico.
6. Alto rendimiento.

Java se ejecuta en más de 850 millones de ordenadores personales en todo el mundo y en millones de dispositivos, como televisiones o dispositivos móviles.

2.1.2.2 C

C [13] es conocido como el lenguaje de programación de sistemas por excelencia, ya que es un lenguaje de programación que no está asociado a ningún sistema operativo, ni a ningún ordenador en especial.

Entre las características de C destacan:

1. Es un lenguaje simple.
2. Permite trabajar en módulos.
3. Trabaja con librerías de funciones.
4. Lenguaje seguro.

El lenguaje de programación C es eficiente, potente, eficaz, rápido e indispensable para cualquier programa.

2.1.2.3 C++

Es un lenguaje de programación orientado a objetos derivado de C. C++ [8] nació para añadirle cualidades y características de las que carecía C, como son los mecanismos que permiten la manipulación de objetos.

Las características más destacadas de C++ son:

1. Orientado a objetos.
2. Lenguaje estructurado.
3. Trabaja con librerías de funciones.
4. Lenguaje seguro.
5. Permite trabajar en módulos.

Actualmente, C++ es el lenguaje de programación utilizado en numerosas bibliotecas multiplataforma para el desarrollo de interfaces gráficas de usuario. Una de las más populares en la actualidad es QT[5,6].

Figura 5: Logotipo de QT

2.1.2.4 Comparativa

A continuación, se va a realizar una comparativa de los lenguajes de programación mencionados anteriormente, mostrando las ventajas y desventajas de cada uno de ellos:

	Java	C	C++
Orientación a objetos	Sí	No	Sí
Portable	Sí	No	No
Rapidez	Media	Alta	Alta
Robustez	Muy alta	Alta	Alta
Multiplataforma	Sí	Sí	Sí
Liberación de memoria	Recolector de basura	Free	Delete

Tabla 2: Comparativa de Lenguajes de programación

2.1.3 Sistemas de control de versiones

Se denomina sistema de control de versiones a la implementación en software de una herramienta que permita automatizar las tareas de guardar, registrar, identificar y mezclar versiones de archivos. Estos sistemas son útiles para archivos que son modificados frecuentemente, como son los programas informáticos.

Las características principales de los sistemas de control de versiones son las siguientes:

1. Respaldo y restauración de archivos versionados.
2. Sincronización de la última versión de un código fuente.
3. Deshacer los cambios retornando a la última versión almacenada en un sistema de control de versiones.
4. Seguimiento de cambios (cambios efectuados y responsable del cambio).
5. Estandarización del código.

Los sistemas de gestión más conocidos en la actualidad son entre otros: CVS, Subversion y Git.

La aplicación a desarrollar está muy relacionada con los sistemas de control de versiones. La relación con estos sistemas se debe a que la aplicación permitirá gestionar y registrar información sobre las versiones de los archivos, más concretamente, sobre las versiones de los sistemas que se encuentran instalados en los aeropuertos que el departamento controla.

2.1.3.1 CVS

CVS [15], también conocido como Concurrent Versioning System, es una herramienta de control de versiones que permite mantener el registro de todo el trabajo y los cambios en los ficheros que forman un proyecto software. También permite que distintos desarrolladores colaboren. Es uno de los sistemas de control de versiones más conocido actualmente ya que se trata de un sistema de software libre.

Las principales características de esta herramienta son:

1. Utiliza una arquitectura cliente-servidor.
2. Es una herramienta multiplataforma.
3. Es una herramienta de software libre.
4. Permite tener archivos de registro.

Desarrollado por GNU, actualmente CVS tiene gran cantidad de versiones implantadas en los diferentes sistemas operativos.

2.1.3.2 Subversion

Este sistema de control de versiones fue diseñado específicamente para poder reemplazar al mencionado anteriormente sistema CVS.

Es un sistema de software libre al igual que CVS, y tiene como principales características:

1. Permite seguir la historia de los archivos y directorios gracias a copias y renombrados.
2. Permite el manejo eficiente de archivos binarios.
3. Las modificaciones realizadas son atómicas, es decir, ante un fallo del sistema, la modificación no puede quedarse a medias.
4. La creación de etiquetas y ramas es una operación más eficiente.

Hoy en día, Subversion [16] es uno de los sistemas más conocidos en la comunidad de software libre y es utilizado en gran cantidad de proyectos.

2.1.3.3 Git

Diseñado por Linus Torvald, es un software de control de versiones pensado para el mantenimiento de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Las características más destacadas de Git [17] son:

1. Permite la gestión eficiente de proyectos grandes.
2. Los renombrados se trabajan basándose en similitudes entre ficheros.
3. Resulta más complicado el trabajar con ficheros concretos frente al trabajo de proyectos.
4. Re-almacenamiento periódico en ficheros, lo que da lugar a una gran eficiencia para escritura de cambios.

El mantenimiento software de Git se realiza en la actualidad mediante la colaboración de unos 280 programadores.

2.1.3.4 Comparativa

A continuación, se va a realizar una comparativa de los sistemas de control de versiones mencionados anteriormente, mostrando las ventajas y desventajas de cada uno de ellos:

	CVS	Subversion	Git
Multiplataforma	Sí	Sí	Sí
Licencia	Gratuita	Gratuita	Gratuita
Orientación	Ficheros	Proyectos	Proyectos
Soporte Unicode	Limitado	Ilimitado	Limitado
Rapidez	Media	Alta	Media
Renombrar / Eliminar	No da soporte	Soporte parcial	Da soporte

Tabla 3: Comparativa de Sistemas de Control de Versiones

2.2 MARCO REGULADOR Y RESTRICCIONES

En este apartado se recogen las reglas y normas que afectan al proyecto que se está realizando.

Principalmente, dicho proyecto está afectado por un conjunto de reglas internas a todos los departamentos de ATM de INDRA que se encuentran recogidas en el documento *IP-ID-5653 Estilo de programación en lenguaje C/C++.*

En este documento se recogen una serie de reglas a seguir a la hora de diseñar y codificar programas escritos en lenguaje C/C++, con el fin de obtener uniformidad y consistencia a lo largo del código generado.

Las normas de programación a seguir a la hora de escritura de código y que deben ayudar a que el programa sea legible y mantenible son:

1. Comentar el código.
2. Utilizar identificadores con prefijos.
3. Estructurar y modularizar el programa.
4. Dar un estilo y apariencia homogénea al código.
5. Cuidar las prácticas de programación.

Por otro lado, los comentarios deben ser utilizados para dar información al lector o para clarificar secciones del código con vistas a su mantenimiento o reutilización. Los comentarios necesitan mantenimiento al igual que el código, por lo que deben de ser comentarios precisos y concisos. Existen 4 tipos de comentarios:

- Cabecera de fichero: para identificar y describir el fichero fuente, y las funciones que lo contienen.

```
/*****************************************************************************
*                                                                            *
*   PROGRAMA       : ./central/src/gestión.c                                 *
*   REALIZADO POR  : J. Victor Lerga Bezunartea                              *
*   FECHA          : 05/04/94                                                *
*   VERSIÓN        : 1.0                                                     *
*                                                                            *
*****************************************************************************
*                           CONTENIDO                                        *
* Este fichero contiene la gestión de mensajes recibidos de una posición     *
* (clase posición::)                                                         *
*                                                                            *
* FuncionAi()                                                                *
* CódigoAi()                                                                 *
* ConexPosPabx()                                                             *
*                                                                            *
*****************************************************************************
*                           MODIFICACIONES                                   *
*                                                                            *
*   VERSIÓN   FECHA     DESCRIPCIÓN                              AUTOR       *
*----------------------------------------------------------------------------*
*     1.0     05/04/94  Versión inicial                          J.V.L.B.    *
*                                                                            *
*****************************************************************************/
```

Figura 6: Cabecera Fichero

- Cabecera de función: permite identificar la función.

```
/*****************************************************************************
*                    FunciónAi(p_tabla,codigo)                                *
*****************************************************************************
*                                                                           *
*  Realiza el tratamiento de entrada de una tecla de función de Acceso Indirec *
*  Tanto la tecla de AI como un código de marcación (número )               *
*                                                                           *
*                        PARÁMETROS ENTRADA                                 *
*                                                                           *
*  p_tabla: Direcc. del ISDN de la  posición que recibe el mensaje en  tabla.  *
*  código:  Código de tecla (CODTEC) en el mensaje recibido.                *
*                                                                           *
*                        PARÁMETROS SALIDA                                  *
*                                                                           *
*                                                                           *
*                                                                           *
*****************************************************************************
*                        MODIFICACIONES                                     *
*                                                                           *
*   VERSIÓN   FECHA     DESCRIPCIÓN                              AUTOR      *
*---------------------------------------------------------------------------*
*     1.0    18/02/94   Versión inicial                          J.V.L.B.   *
*****************************************************************************/
```

Figura 7: Cabecera Función

- Comentarios explicativos del código.

Si no cabe en 80 caracteres, deberá ser:

```
/* Esto es un comentario para documentación. Este comentario tiene una
longitud superior a 80 caracteres y por ello está partido en varias
líneas */
```

No deberá ser:

```
/*Este comentario tiene una longitud superior a 80 caracteres y por ello */
/*está partido en varias líneas, pero esto no está permitido porque      */
/*cada línea es un nuevo comentario                                       */
```

No deberá ser:

```
//Este comentario tiene una longitud superior a 80 caracteres y por ello
//está partido en varias líneas, pero esto no está permitido porque
//cada línea es un nuevo comentario
```

Figura 8: Comentario Explicativo

- Comentarios de cambios de software.

```
/*** VLB 22/02/01 3R6 Se encontró un error en las PAF respecto a la
ejecución de la llamada  **/
/*** VLB 22/02/01 3R6 eliminado llamada = llamada.telefonía[0]; **/
llamada = llamada.telefonía[1];
llamada += llamada.líneaCaliente[1];
/*** VLB 22/02/01 3R6 fin modificación  **/
```

Figura 9: Comentario Cambio Software

Una estructura adecuada de programa aumenta la claridad, por lo que también hay una serie de reglas que afectan a las clases. Estas son:

- Sólo se permitirá una única declaración de clase visible en cada fichero de cabecera.
- En la cabecera de la clase se declararán, en este orden, si existen, las partes public, protected y private. Se intentará definir las interfaces dejando lo mínimo público y el resto protegido o privado.
- Los ficheros de cabecera no podrán contener definiciones de funciones, sino sólo declaraciones.
- Evitar la existencia de datos compartidos. Han de quedar encapsulados en clases.

Por último, se consideran una serie de reglas y recomendaciones a tener en cuenta en los prefijos e identificadores. Se puede observar en la siguiente tabla:

Identificador	Descripción
Constantes	**XXXX_YYYY** Se describen en mayúsculas y nombres separados por "_". #define **NUM_PAG** 12
Tipos	**T_XXX_YYY** Se definen en mayúsculas con una "T" como primer carácter. p.ej. **T_TABLA_NOMBRES**
Clases	**C_XxxxYyyy** Se definen con el prefijo "C_" y las palabras que componen el nombre comenzarán por mayúsculas. p.ej. **C_PosiciónRadio**
Variables	**xxxYyyy** Se definen con la primera palabra en minúsculas, y el resto comenzando por mayúscula. p. ej. **datos Posición, tabla,**

Campos de Enumerados	**E_XXXX** Se definen en mayúsculas, con una “E” como primer carácter. p. ej. **E_OK, E_ERROR**
Funciones o métodos	**XxxxYyyy()** Se definen con mayúsculas en la primera letra de cada palabra. p. ej. **FunciónNueva1()**
Variables miembro de una clase	**m_xxxxxYxxxx** Se definen con el prefijo “m_” y el nombre de la variable siguiendo la norma de definición de variables. p. ej. **M_datosPosición**
Punteros	**pxxxYyyy** Se definen con el prefijo “p” y el nombre de la variable a que apuntan. p. ej. **pdatosPosición, ptabla**

Tabla 4: Prefijos en identificadores

3. ANÁLISIS Y DISEÑO DE LA SOLUCIÓN TÉCNICA

3.1 METODOLOGÍA

Las metodologías de desarrollo software surgen con la necesidad de estructurar, planificar y controlar el proceso de desarrollo de un producto software. Podemos distinguir dos tipos de metodologías [18]:

- Metodologías ágiles.
- Metodologías tradicionales.

En la siguiente tabla se resumen las principales diferencias entre ambas metodologías:

Metodologías ágiles	Metodologías tradicionales
Basadas en procedimientos provenientes de prácticas de producción de código. Diseñadas para cambios durante el trascurso del proyecto.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo. Ofrecen resistencia a los cambios en el trascurso del proyecto.
Proceso menos controlado, con apenas normas.	Proceso mucho más controlado, con numerosas normas.
No existe contrato prefijado. Si lo hubiera, es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños que trabajan en el mismo sitio (menos de 10 trabajadores)	Grupos grandes y se encuentran distribuidos.
Pocos roles.	Muchos roles.
Menos énfasis en la arquitectura software.	La arquitectura software es esencial.

Tabla 5: Comparativa de tipos de metodología

En el presente capítulo, se expondrá la metodología empleada para la realización del proyecto y se justificará la elección de la misma.

3.1.1 Metodologías ágiles

Las metodologías ágiles han tenido una mayor aceptación que las metodologías tradicionales entre los desarrolladores de proyectos. Esto se debe a la simplicidad de las reglas, la orientación que tienen estas metodologías hacia proyectos formados por equipos de desarrollo pequeños y la flexibilidad ante los cambios, entre otros factores mencionados anteriormente.

El proyecto ha sido desarrollado mediante el empleo de una metodología ágil. Es un proyecto que a lo largo del proceso de desarrollo sufrirá cambios, por lo que para ello es conveniente el empleo de una metodología ágil, una metodología que se adapta.

Dentro de las metodologías ágiles, podemos distinguir varios tipos: SCRUM, *Crystal Methodologies*, *Dynamic Systems Development Method* (DSDM), *Adaptative Software Development* (ASD), etc.

La metodología ágil empleada en el presente proyecto es *Adaptative Software Development* [19]. Impulsada por Jim Highsmith, la ASD se caracteriza por ser iterativa, orientado a los componentes software y tolerante a los cambios.

3.1.1.1 ASD: Ciclo de vida

El ciclo de vida propuesto tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la fase de especulación, se inicia el proyecto y se planifican las características del software. En la segunda fase, la fase de colaboración, se desarrollan las características planificadas en la primera fase. Por último, en la fase de aprendizaje, se revisa la calidad del proyecto y se entrega al cliente. Gracias a la revisión, se puede aprender de los errores y así volver a iniciar el ciclo de desarrollo.

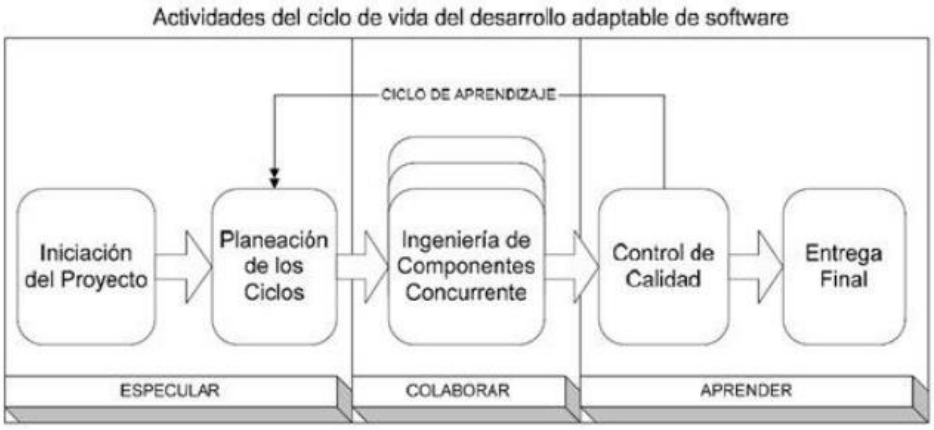


Figura 10: Ciclo de vida ASD

El presente proyecto, tendrá un ciclo de vida con una única iteración, dejando para una segunda iteración las mejoras planteadas durante el control de calidad.

A continuación, se muestran las tareas que se realizarán en esta primera iteración:

Fase de especulación

Como se ha mencionado anteriormente, en esta fase se inicia el proyecto y se planifican las características del software.

- Definición de requisitos.
- Características software del proyecto.

- Planificación temporal. Se encuentra recogido en el diagrama de Gantt del apartado 8.1.

Fase de colaboración

En esta fase se lleva a cabo el desarrollo de lo planificado durante la especulación.

- Descripción / Diseño del sistema. Se realizará una descripción de la aplicación del proyecto y las funcionalidades que tendrá en función de los requisitos trazados.
- Implementación. Se implementará un prototipo de la aplicación diseñada y analizada.

Fase de aprendizaje

En esta última fase, se analiza la calidad de la aplicación desarrollada y se entrega al cliente.

- Rendimiento y calidad. Se lleva a cabo una evaluación de la aplicación.
- Mejoras y errores. Se detectan mejoras y errores para corregir en una segunda iteración.
- Conclusiones del proyecto.

3.2 ANÁLISIS DEL SISTEMA

En este apartado se llevará a cabo una descripción detallada sobre el resultado esperado del sistema. Se explicarán los casos de uso para los cuales se utilizará la aplicación desarrollada y por último, se expondrán de manera clara y precisa los requisitos y funcionalidades que tendrá el sistema.

3.2.1 Casos de uso

Los casos de uso especifican el comportamiento del sistema desde el punto de vista del usuario. Es decir, describen lo que hace el sistema, pero no el cómo lo hace. Los casos de uso del sistema se pueden observar en la figura 20:

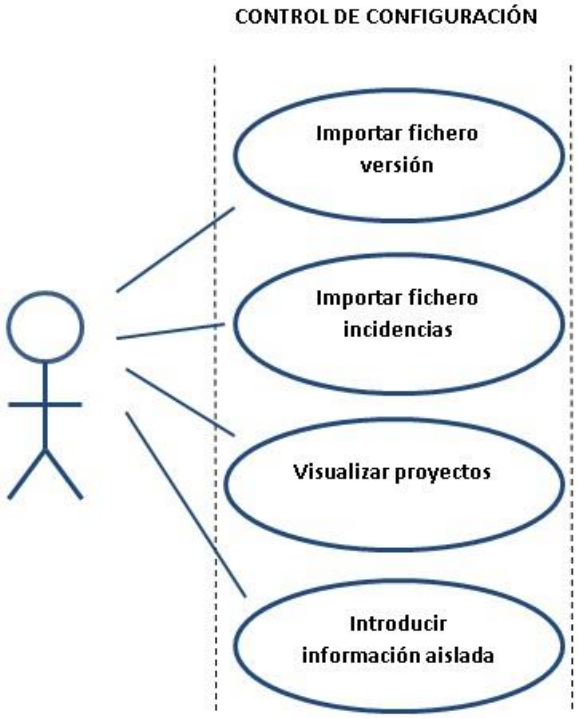


Figura 11: Casos de uso

En la figura podemos observar el nivel más general de abstracción de los casos de uso. Internamente la aplicación tiene gran cantidad de funcionalidades, pero de manera general y simplificada, los casos de uso del sistema para el usuario son esos cuatro y coinciden prácticamente con los objetivos del proyecto. Cabe destacar que, en el sistema no habrá distinción de usuarios, es decir, hay un solo tipo de usuario, o en otras palabras, cualquier usuario que acceda a la aplicación puede realizar las mismas funciones.

3.2.2 Requisitos software

Como se ha comentado anteriormente, el presente proyecto fin de grado consiste en la implementación de un sistema para gestión de versiones software basado en incidencias. Para poder implementar este sistema, se necesita establecer una serie de requisitos software que servirán de base para saber cómo será finalmente la aplicación.

Los requisitos software se dividen en dos grupos fundamentalmente: los requisitos funcionales y los requisitos no funcionales.

3.2.2.1 Requisitos funcionales

Los requisitos funcionales definen el comportamiento del software del sistema. Para su definición, los requisitos funcionales se componen de los siguientes campos:

- Identificador: está compuesto por las iniciales "RSF" seguidas del número del requisito.
- Descripción: contiene la explicación del requisito.
- Necesidad: describe si el requisito es esencial u opcional.
- Prioridad: describe el grado de prioridad de implementación del requisito. Puede ser Baja, Media o Alta.

En el caso de la aplicación desarrollada, la mayoría de los requisitos funcionales están relacionados con la interfaz gráfica:

RSF-01			
Descripción		El sistema deberá mostrar inicialmente una ventana de login.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 6: RSF-01 Ventana de Login

RSF-02			
Descripción		La ventana de login constará de un campo para introducir el usuario, otro campo para introducir la contraseña, un botón de "Aceptar" y otro de "Cancelar".	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 7: RSF-02 Ventana de Login

RSF-03			
Descripción		El sistema deberá permitir al usuario salir de la aplicación si pulsa el botón Cancelar.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 8: RSF-03 Ventana de Login

RSF-04			
Descripción		El sistema deberá permitir al usuario entrar en la aplicación si introduce correctamente el usuario y contraseña, y pulsa el botón Aceptar.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 9: RSF-04 Ventana de Login

RSF-05			
Descripción		El sistema deberá mostrar una ventana principal tras el login. Esta ventana está compuesta por un menú superior, un árbol donde se muestran los proyectos, una tabla de componentes y una tabla de incidencias.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 10: RSF-05 Ventana principal

RSF-06			
Descripción		El menú superior estará compuesto por las opciones “Archivo” y “Editar”.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 11: RSF-06 Opciones menú

RSF-07			
Descripción		La opción “Archivo” del menú está compuesta por “Importar Versión Sistema”, “Importar Incidencias” y “Salir”.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 12: RSF-07 Opciones menú

RSF-08			
Descripción		La opción “Editar” del menú está compuesta por “Proyectos”, “Versiones”, “CSCIs”, “HWCIs”, “Componentes” e “Incidencias”.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 13: RSF-08 Opciones menú

RSF-09			
Descripción		Si el usuario selecciona la opción “Importar Versión Sistema”, aparecerá una nueva ventana que permitirá realizar la acción.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 14: RSF-09 Opciones menú

RSF-10			
Descripción		Si el usuario selecciona la opción “Importar Incidencias”, aparecerá una nueva ventana que permitirá realizar la acción.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 15: RSF-10 Opciones menú

RSF-11			
Descripción		Si el usuario selecciona la opción “Salir”, se cerrará la aplicación.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 16: RSF-11 Opciones menú

RSF-12			
Descripción		Si el usuario selecciona alguna de las opciones del menú “Editar”, el sistema mostrará una ventana en función de la opción seleccionada y permitirá al usuario introducir manualmente el elemento correspondiente a la opción seleccionada.	
Necesidad	Opcional	**Prioridad**	Media

Tabla 17: RSF-12 Opciones menú

RSF-13			
Descripción		La ventana de “Importar Versión Sistema” debe contener un botón que permita explorar la ruta del fichero csv, campos que permitan una visualización previa del fichero cargado y asignar un emplazamiento. Un botón “Importar” y un botón “Cancelar”.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 18: RSF-13 Importar Versión Sistema

RSF-14			
Descripción		El botón "Importar" se encuentra deshabilitado por defecto. Cuando se carga un fichero, el botón de "Importar" se habilitará, permitiendo realizar la acción en el sistema.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 19: RSF-14 Importar Versión Sistema

RSF-15			
Descripción		Si el usuario pulsa el botón "Cancelar", la ventana de "Importar Versión Sistema" se cerrará. También tiene el mismo comportamiento en "Importar Incidencias".	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 20: RSF-15 Importar Versión Sistema

RSF-16			
Descripción		Si el fichero importado pertenece a un proyecto que no tiene ningún emplazamiento asignado, el sistema informará de ello y creará un emplazamiento por defecto.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 21: RSF-16 Importar Versión Sistema

RSF-17			
Descripción		El sistema comprobará que el fichero importado no esté duplicado, evitando almacenar información repetida en el mismo.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 22: RSF-17 Importar Versión Sistema

RSF-18			
Descripción		La ventana de "Importar Incidencias" debe contener un botón que permita explorar la ruta del fichero csv, además de un botón "Importar" y un botón "Cancelar".	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 23: RSF-18 Importar Incidencias

RSF-19			
Descripción		Si el usuario selecciona un proyecto del árbol de proyectos, el sistema deberá mostrar la lista de emplazamientos asociados a ese proyecto.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 24: RSF-19 Árbol de proyectos

RSF-20			
Descripción		Si el usuario selecciona un emplazamiento que esté asociado a un proyecto, el sistema deberá mostrar la lista de versiones del sistema que hay en ese emplazamiento.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 25: RSF-20 Árbol de proyectos

RSF-21			
Descripción		Si el usuario selecciona una versión del sistema perteneciente a un emplazamiento, el sistema deberá mostrar en la tabla de componentes, la lista de componentes que pertenezcan a esa versión.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 26: RSF-21 Tabla de componentes

RSF-22			
Descripción		Si algún componente tiene asociado alguna incidencia, estas serán mostradas en la tabla de incidencias cuando el usuario seleccione el componente. En el caso de que el componente no tenga ninguna incidencia asociada, no se mostrará nada en la tabla de incidencias.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 27: RSF-22 Tabla de incidencias

3.2.2.2 Requisitos no funcionales

Los requisitos no funcionales imponen restricciones en el diseño o la implementación del sistema. Para su definición, los requisitos no funcionales se componen de los siguientes campos:

- Identificador: está compuesto por las iniciales “RSNF” seguidas del número del requisito.

- Descripción: contiene la explicación del requisito.
- Necesidad: describe si el requisito es esencial u opcional.
- Prioridad: describe el grado de prioridad de implementación del requisito. Puede ser Baja, Media o Alta.

RSNF-01			
Descripción		El sistema tendrá la interfaz en castellano.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 28: RSNF-01 Interfaz de usuario

RSNF-02			
Descripción		El sistema será robusto frente a posibles errores.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 29: RSNF-02 Rendimiento del sistema

RSF-21			
Descripción		La aplicación tiene que ser mantenible, mediante la escritura de un código limpio y comentado.	
Necesidad	Esencial	**Prioridad**	Alta

Tabla 30: RSNF-03 Mantenibilidad de la aplicación

4. DISEÑO

El presente proyecto fin de carrera está muy relacionado con los sistemas de control de versiones, ya que consiste en la realización de una aplicación para llevar el control del versionado software y hardware de los sistemas de control de tráfico aéreo del departamento. La aplicación controlará y gestionará las versiones del sistema que se encuentren instaladas en los diferentes aeropuertos, junto con el versionado de los ficheros de configuración para dichas versiones del sistema.

El departamento lo que necesita es una aplicación sencilla que puedan utilizar todos los miembros del SCV. Una herramienta simplemente de consulta y de gestión de información, donde no sea necesario subir código a ningún repositorio. En esto se diferencia principalmente de los sistemas de control de versiones, ya que la aplicación a desarrollar no se encargará del código, si no que se encargará de almacenar y gestionar la información acerca de las versiones de los sistemas. Una herramienta que permita de manera rápida y eficaz saber la información relativa a un proyecto concreto.

Para llevar a cabo el diseño del sistema que gestione la totalidad de versiones que controla el departamento SCV, se debe distinguir entre dos partes fundamentales:

- **Diseño del núcleo**. El núcleo del sistema está formado principalmente por el diseño de la base de datos que almacenará toda la información sobre versiones, componentes, incidencias, proyectos, etc.

	A	B	C	D	E	F	G	H	I	J	K
1	Proyecto	Emplazamiento	VersionSistem	Build	Patch	NombreVersion	FechaInstalaci	EnviadaClier	sdcxxi	FechaSdcxxi	openh323
2	Arabia		2.1.05	05	00	SDCXXI_HMI_2000 (Turquia)	23/12/2011	VERDADERO	6.16.41.1022	23/12/2011	22.2.2.8
3	Arabia		2.1.05	06	03	SDCXXI_HMI_2000 (Turquia)	25/05/2012	VERDADERO	6.16.41.1022	23/12/2011	22.2.2.8
4	Argentina	Cordoba	1.1.01	03	00	SDCXXI_HMI_COPERNICUS I/II	14/02/2011	VERDADERO	6.12.0.3042	19/07/2010	20.6.1.0
5	Argentina	Cordoba	1.1.01	06	00	SDCXXI_HMI_COPERNICUS I/II	02/11/2011	VERDADERO	6.12.0.3045	21/02/2011	20.6.1.0
6	Argentina	Cordoba	1.1.01	07	03	SDCXXI_HMI_COPERNICUS I/II	27/04/2012	VERDADERO	6.12.0.3053	11/04/2012	22.2.2.8
7	Argentina	Cordoba	1.1.01	07	06	SDCXXI_HMI_COPERNICUS I/II	28/08/2012	VERDADERO	6.12.0.3053	11/04/2012	22.2.2.8
8	ARS-D	Sevilla	1.2.00	01	00	SDCXXI_ARSD_HMI_MILITAR	03/02/2012	FALSO	6.8.59.536	02/02/2012	12.3.11.0
9	ARS-D	Sevilla	1.2.00	01	01	SDCXXI_ARSD_HMI_MILITAR	10/02/2012	VERDADERO	6.8.59.536	09/02/2012	12.3.11.0
10	Azerbaiyán	Bakú	2.1.05	06	04	SDCXXI_HMI_2000 (Turquia)	04/06/2012	FALSO	6.16.41.1022	23/12/2011	22.2.2.8
11	Azerbaiyán	Bakú	2.1.05	07	11	SDCXXI_HMI_2000 (Turquia)	14/03/2013	FAT	6.16.41.1041	22/11/2012	24.5.0.0
12	Bosnia		2.1.00	01	00	SDCXXI_HMI_2000 (Bosnia)	28/10/2011	VERDADERO	6.16.2.1050	29/09/2010	20.6.1.0
13	Bosnia		2.1.00	03	00	SDCXXI_HMI_2000 (Bosnia)	18/01/2011	VERDADERO	6.16.2.1050	29/09/2010	20.6.1.0
14	Bosnia		2.1.00	05	00	SDCXXI_HMI_2000 (Bosnia)	22/02/2011	VERDADERO	6.16.2.1070	28/03/2011	22.2.2.0
15	Bosnia		2.1.00	13	00	SDCXXI_HMI_2000 (Bosnia)	20/07/2012	FALSO	6.16.2.1128	03/07/2012	24,0,0,0
16	Bosnia		2.1.00	13	01	SDCXXI_HMI_2000 (Bosnia)	23/07/2012	FALSO	6.16.2.1131	20/07/2012	24,0,0,0
17	Bosnia		2.1.00	13	02	SDCXXI_HMI_2000 (Bosnia)	30/07/2012	FALSO	6.16.2.1134	30/07/2012	24,0,0,0
18	Bosnia		2.1.00	13	03	SDCXXI_HMI_2000 (Bosnia)	31/07/2012	FALSO	6.16.21.1000	31/07/2012	24,0,0,0
19	Bosnia		2.1.00	13	04	SDCXXI_HMI_2000 (Bosnia)	02/08/2012	FALSO	6.16.21.1001	01/08/2012	24,0,0,0
20	Bosnia		2.1.00	13	05	SDCXXI_HMI_2000 (Bosnia)	21/08/2012	FALSO	6.16.21.1001	01/08/2012	24,0,0,0
21	Bosnia		2.1.00	13	06	SDCXXI_HMI_2000 (Bosnia)	21/08/2012	FALSO	6.16.21.1001	01/08/2012	24,0,0,0
22	Bosnia		2.1.00	13	09	SDCXXI_HMI_2000 (Bosnia)	12/09/2012	FALSO	6.16.21.1004	24/08/2012	24,0,0,0

Figura 12: Excel Control de Configuración

- **Diseño de la interfaz gráfica de usuario (GUI).** En el diseño de la GUI podemos distinguir entre cuatro bloques clave:

    - **Diseño de la persistencia con la base de datos.** La GUI necesita conectarse a la base de datos para que toda la información introducida por el usuario sea almacenada en la misma.

- **Diseño de los metadatos.** Se encargan de almacenar o cargar la información en la base de datos.
- **Diseño de los servicios.** Los servicios son el enlace entre la capa de negocio (HMI) y los metadatos.
- **Diseño del HMI.** Diseño gráfico de la aplicación para que el usuario pueda comunicarse con la misma.

A continuación, en el presente capítulo, se explicará el diseño de cada una de las partes enumeradas.

4.1 DISEÑO DEL NÚCLEO

Como se ha mencionado anteriormente, el núcleo del sistema lo constituye la base de datos diseñada para almacenar toda la información del sistema.

La base de datos contiene toda la información importante y necesaria que tiene que ser controlada a la hora de revisar alguna versión del sistema de comunicación de voz digital de algún aeropuerto. Es decir, si un instalador se dispone a desplegar una versión en un aeropuerto, toda la información que necesita es la que tiene que estar almacenada en la base de datos diseñada.

- **Proyectos.** La base de datos almacenará la información relativa a cada uno de los proyectos para los que se encuentra trabajando el departamento SCV.

  - **Nombre** del proyecto.
  - **Emplazamientos** que tiene el proyecto. Un proyecto, como por ejemplo Ecuador, abarca más de un emplazamiento, Quito y Guayaquil.
  - **Versiones** del sistema que tiene el proyecto. Los emplazamientos que tiene un proyecto no tienen por qué tener las mismas versiones instaladas.

- **Componentes.** La base de datos almacenará la información relativa a un componente software.

  - **Nombre** del componente software.
  - **Versiones** del componente software. Los componentes software varían sus versiones cuando son modificados debido a alguna incidencia.
  - **Checksum** del componente software. Cada versión de un componente software tiene un checksum diferente.
  - Identificación de si es un **COTS**.
  - **HWCI** al que pertenece el componente software. Se necesita localizar el hardware donde va cada componente software.

- **CSCI** al que pertenece el componente software. Se necesita localizar a que familia software dentro de los principales grupos existentes (TMCS, CWP, SeeSCV o GWP) pertenece el componente. Puede pertenecer a más de un grupo dicho componente.

- **Incidencias.** La base de datos también tiene que almacenar la información relativa a las incidencias que haya en las versiones del sistema y debido a las cuales se generan otras versiones nuevas.

    - **Identificador** de la incidencia. Las incidencias llevan un identificador formado por la base de datos a la que pertenece la incidencia y por un número (SCV#1342, ISCV#2345, etc.).
    - **Synopsis** de la incidencia.
    - **Descripción** de la incidencia.
    - **Tipo** de incidencia. Las incidencias pueden ser hardware o software.
    - **Estado** de la incidencia. Las incidencias pueden estar en concluded o in acceptance test.
    - **Base de datos** a la que pertenece la incidencia (interna o externa).

4.1.1 Estructura modelo relacional

La base de datos ha sido implementada mediante PostgreSQL y la estructura de la misma y cómo se relacionan sus campos y tablas se puede observar a continuación en la figura 21:

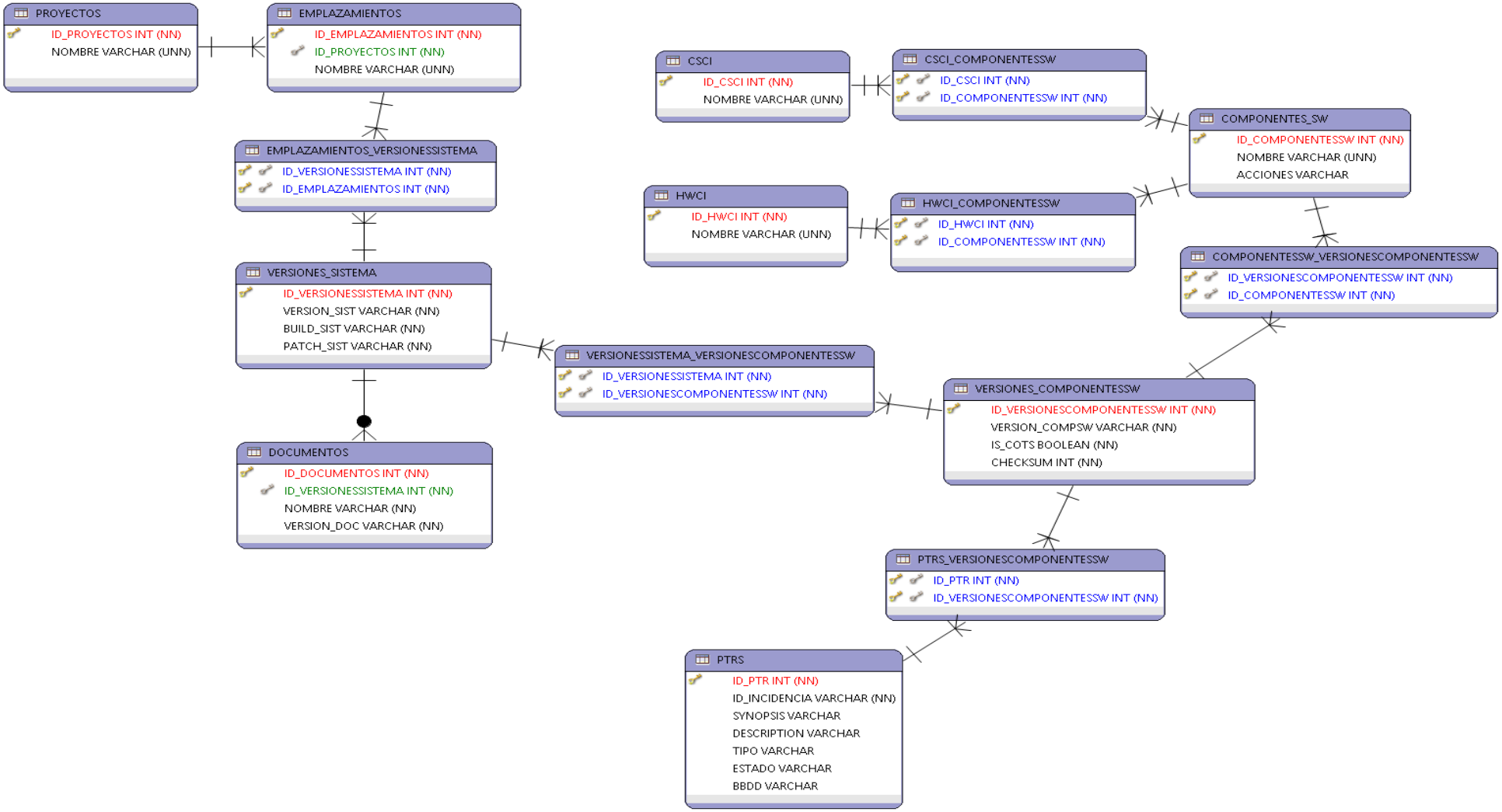


Figura 13: Modelo relacional

Como se puede observar en el modelo relacional, la base de datos está formada por:

- Una tabla **Proyectos,** donde se guarda el nombre de cada proyecto.
- Una tabla **Emplazamientos,** donde se guardan los nombres de los emplazamientos que puede tener un proyecto.
- Una tabla **Versiones_Sistema,** donde se guarda la información relativa a una versión de un sistema y que se relaciona con sus respectivos emplazamientos gracias a la tabla **Emplazamientos_VersionesSistema.**
- Una tabla llamada **Documentos,** para guardar información sobre algún documento de instalación que tenga una versión del sistema.
- Una tabla **CSCI,** que almacena la información relativa a los CSCIs.
- Una tabla **HWCI,** que almacena la información relativa a los HWCIs.
- La tabla **Componentes_Sw,** que se encarga de almacenar toda la información relativa a los componentes software. Cada componente software se relaciona con su correspondiente CSCI y con su correspondiente HWCI gracias a las tablas **CSCI_ComponentesSw** y **HWCI_ComponentesSw.**
- La tabla **Versiones_ComponentesSw,** almacena la información relativa a todas las versiones que tiene un componente, relacionándola gracias a la tabla **ComponentesSw_VersionesComponentesSw.** Esta tabla es la que permite relacionar una versión del sistema, con el resto de información de la base de datos gracias a la tabla **VersionesSistema_VersionesComponentesSw.**
- La tabla **Ptrs** almacena la información relativa a las incidencias que haya en una versión. Una incidencia afecta directamente a unos componentes, por lo que se relaciona con estos gracias a la tabla **Ptrs_VersionesComponentesSw.**

4.2 DISEÑO GUI

Una vez diseñada la base de datos en la cual vamos a almacenar toda la información relativa al sistema, el diseño de la interfaz gráfica de usuario será la parte principal de la aplicación.

Para el diseño de la interfaz gráfica, se ha optado por una división en cuatro capas, con el objetivo de que el código sea claro, reutilizable y mantenible: capa de persistencia, capa de metadatos, capa de servicio y capa de negocio.

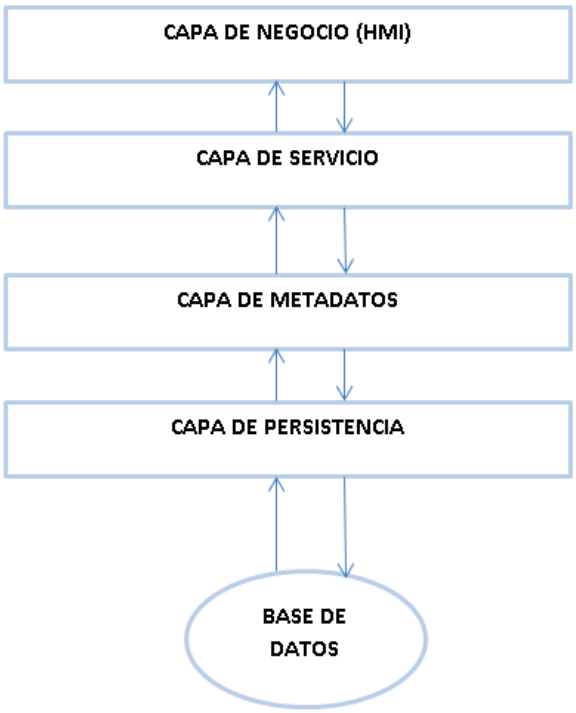


Figura 14: Diseño de capas GUI

4.2.1 Persistencia con base de datos

Para la conexión de la aplicación con la base de datos, QT tiene unos drivers nativos que permiten realizar esta conexión.

La persistencia con la base de datos del sistema se encuentra implementada en dos clases, que son las encargadas de realizar todas las acciones necesarias con la base de datos:

- **C_DbManager**. Esta clase se encarga de realizar la conexión propiamente dicha con la base de datos.
- **C_SqlFactory**. Es la encargada de realizar todas las acciones SQL de la base de datos.

4.2.2 Metadatos

Como se ha comentado anteriormente, la capa de metadatos es con la que se trabaja siempre en el sistema. Nunca se trabaja directamente sobre la base de datos. Todas las operaciones, antes de realizarlas en la base de datos, se realizan previamente en los metadatos.

En la figura 23 se puede observar un ejemplo sobre la utilización de los metadatos en la importación de información por un usuario:

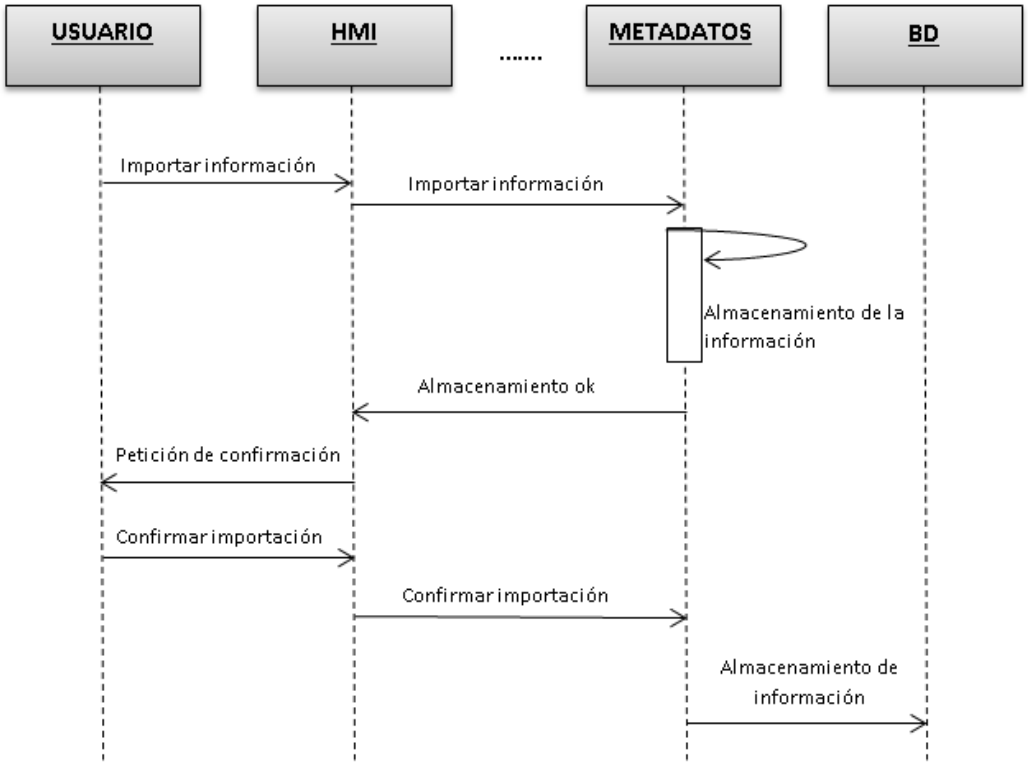


Figura 15: Uso de metadatos en la importación

Como se puede observar en la figura, la información que un usuario introduce en el sistema no es almacenada directamente, sino que es almacenada en los metadatos. Una vez almacenada en los metadatos, el sistema sigue trabajando con la información almacenada en los metadatos, y hasta que el usuario no lo confirme, no se vuelca la información a la base de datos.

Los metadatos se encuentran implementados en las siguientes clases:

- **C_MetadataColumn**. Se encarga de realizar todas las acciones con las columnas.
- **C_MetadataRow**. Se encarga de realizar todas las acciones con las filas.
- **C_MetadataTable**. Se encarga de realizar todas las acciones con las tablas de la base de datos (filas y columnas).
- **C_MetadataTabs**. Esta clase, agrupa las acciones de las tablas de la base de datos. Es un nivel más de abstracción de la clase C_MetadataTable.

Para no tener que trabajar directamente sobre las tablas, columnas y filas de la base de datos, se han implementado metadatos que agrupan funcionalidades y características. Son los siguientes:

- **C_Proyecto**. Agrupa todas las acciones que los metadatos pueden realizar sobre la información relacionada con los proyectos (nombre, emplazamiento, etc.).
- **C_VersionesSistema**. Agrupa todas las acciones que los metadatos pueden realizar sobre la información relacionada con las versiones del sistema (documentos, versiones de componentes, incidencias, etc.).
- **C_Componentes**. Agrupa todas las acciones que los metadatos pueden realizar sobre la información relacionada con los componentes software (CSCI, HWCI, componentes, etc.).

4.2.3 Servicios

Se encargan de conectar/enlazar los metadatos con el HMI, es decir, con el usuario. Los servicios que se han implementado son los siguientes:

- **C_ServiceTablesDefines**. Contiene el nombre de las tablas y las columnas de la base de datos.
- **C_ServiceProyecto**. Agrupa todas las acciones que se pueden realizar sobre los metadatos de un proyecto.
- **C_ServiceVersionesSistema**. Agrupa todas las acciones que se pueden realizar sobre los metadatos de las versiones de un sistema.
- **C_ServiceComponentes**. Agrupa todas las acciones que se pueden realizar sobre los metadatos de un componente software.

4.2.4 HMI

El diseño del HMI da lugar al aspecto visual que va a tener la aplicación, la apariencia que va a tener frente al usuario.

Se ha optado por un aspecto visual sencillo y simple, ya que no era un requisito del sistema. En futuras mejoras, el aspecto visual de la aplicación será uno de los objetivos.

Antes de que el usuario pueda utilizar la aplicación, uno de los requisitos software establecidos es la necesidad de loguearse.

La ventana que la aplicación muestra es la siguiente:

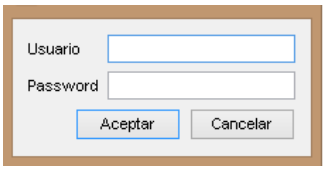


Figura 16: Ventana de login

En primer lugar, hay que hablar de la ventana principal de la aplicación. Es la ventana en la cual el usuario puede visualizar toda la información que necesita sobre algún proyecto del departamento SCV.

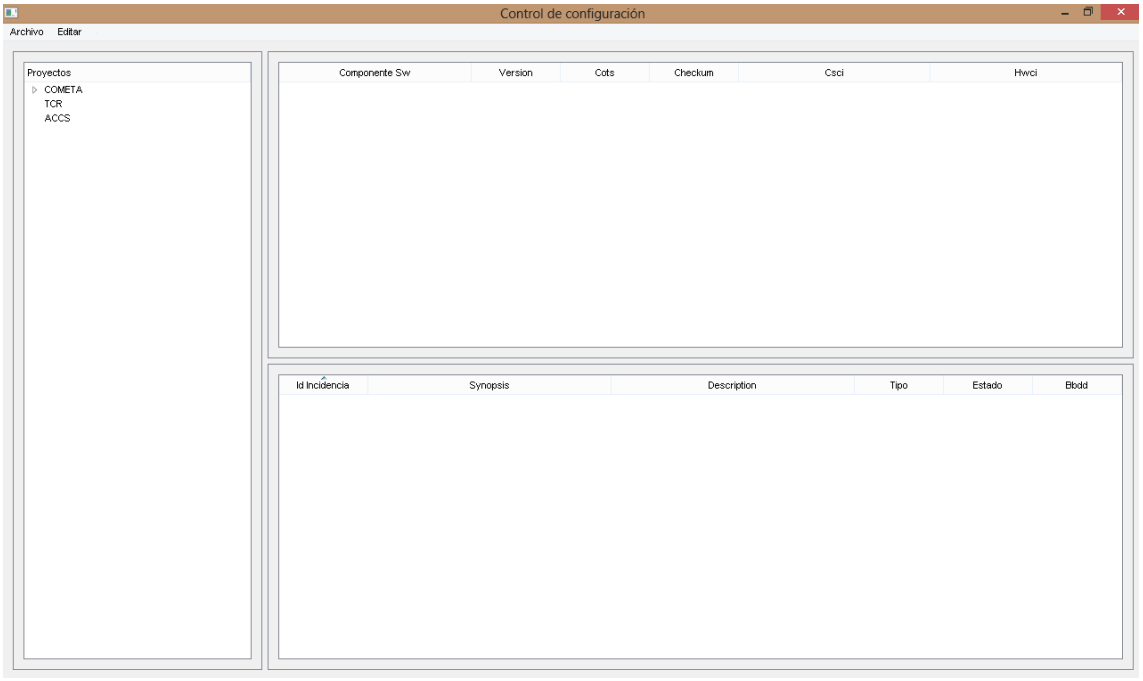


Figura 17: Ventana principal

En la parte superior de la ventana se encuentra un menú con el cual podemos:

- Introducir información aislada de algún proyecto (Editar).
- Importar los ficheros csv con la información relativa a una versión del sistema o salir de la aplicación (Archivo).

Más adelante veremos las opciones mencionadas del menú.

En la parte izquierda de la ventana principal, podemos observar un árbol donde se encuentran todos los proyectos para los cuales tenemos información en el sistema.

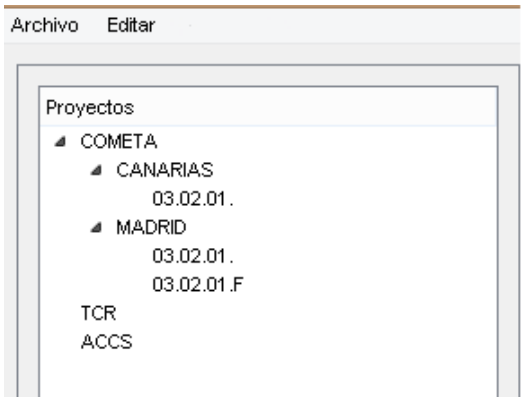


Figura 18: Ventana principal-Árbol de proyectos

Observando la figura anterior, en el caso del proyecto COMETA, tenemos en el sistema información relativa a dos emplazamientos (CANARIAS y MADRID) y cada emplazamiento tiene sus versiones del sistema, que pueden ser la misma (versión 03.02.01.).

Si se hace doble click sobre una versión, aparecerán en la tabla superior de la derecha la lista de componentes que tiene esa versión, con toda la información correspondiente a cada uno de ellos. Se puede observar en la siguiente figura:

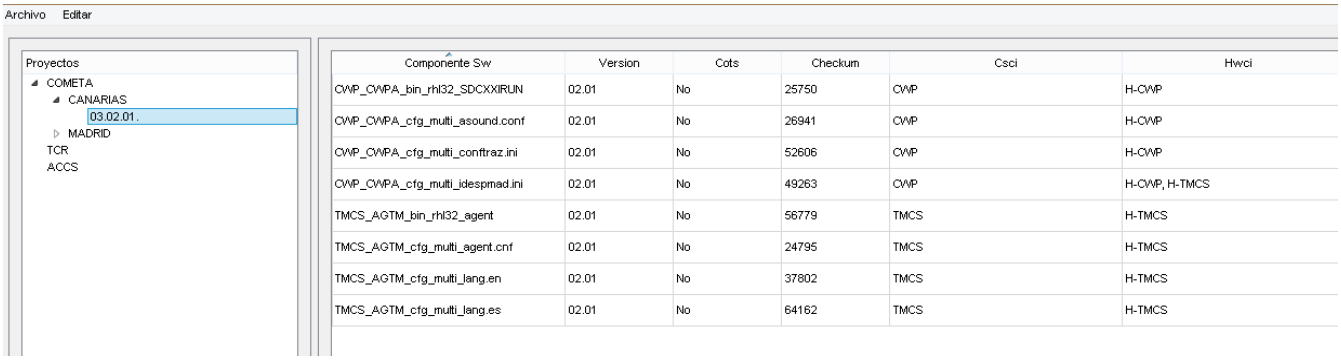
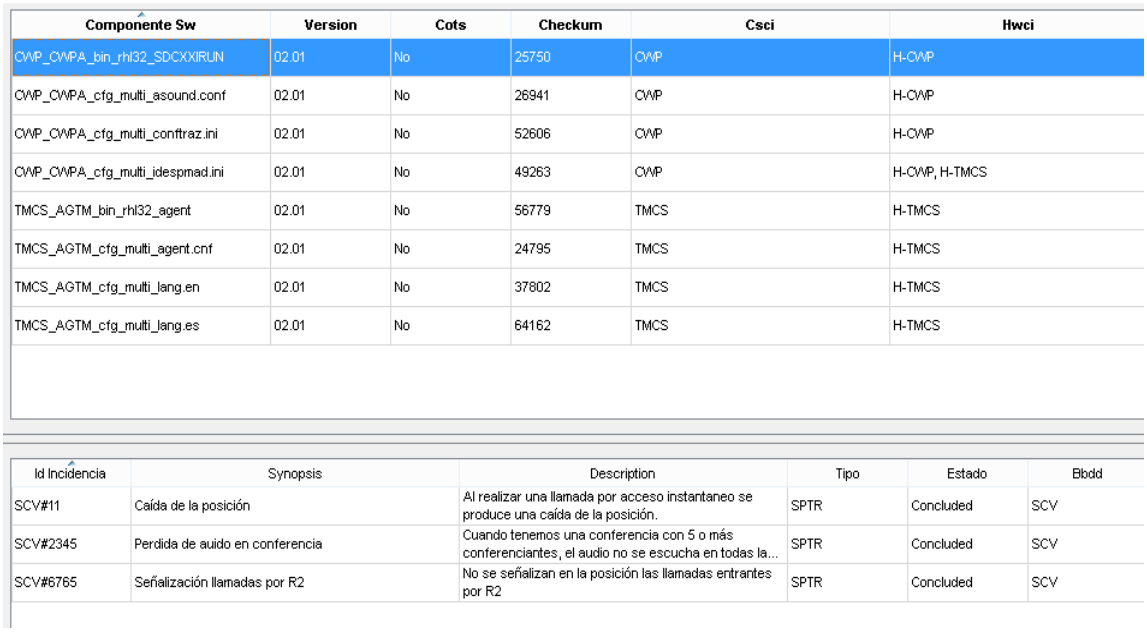


Figura 19: Ventana principal-Tabla de componentes

Si posteriormente realizamos doble click sobre algún componente software, en el caso de que este tenga asociado una o varias incidencias, aparecerán en la tabla de la parte inferior derecha de la ventana principal.



Componente Sw	Version	Cots	Checkum	Csci	Hwci
CWP_CWPA_bin_rhl32_SDCXXIRUN	02.01	No	25750	CWP	H-CWP
CWP_CWPA_cfg_multi_asound.conf	02.01	No	26941	CWP	H-CWP
CWP_CWPA_cfg_multi_conftraz.ini	02.01	No	52606	CWP	H-CWP
CWP_CWPA_cfg_multi_idespmad.ini	02.01	No	49263	CWP	H-CWP, H-TMCS
TMCS_AGTM_bin_rhl32_agent	02.01	No	56779	TMCS	H-TMCS
TMCS_AGTM_cfg_multi_agent.cnf	02.01	No	24795	TMCS	H-TMCS
TMCS_AGTM_cfg_multi_lang.en	02.01	No	37802	TMCS	H-TMCS
TMCS_AGTM_cfg_multi_lang.es	02.01	No	64162	TMCS	H-TMCS

Id Incidencia	Synopsis	Description	Tipo	Estado	Bbdd
SCV#11	Caída de la posición	Al realizar una llamada por acceso instantaneo se produce una caída de la posición.	SPTR	Concluded	SCV
SCV#2345	Perdida de auido en conferencia	Cuando tenemos una conferencia con 5 o más conferenciantes, el audio no se escucha en todas la...	SPTR	Concluded	SCV
SCV#6765	Señalización llamadas por R2	No se señalizan en la posición las llamadas entrantes por R2	SPTR	Concluded	SCV

Figura 20: Ventana principal-Tabla de incidencias

Como se mencionó anteriormente, en el menú superior, una de las acciones que el sistema permite hacer al usuario, es la de introducir información aislada de algún proyecto.

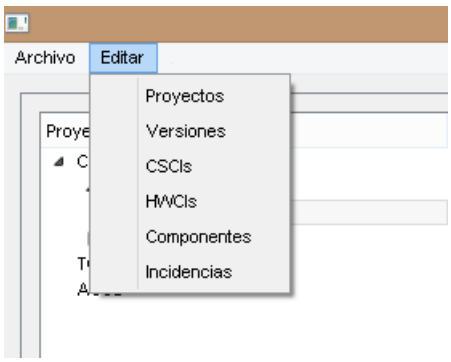


Figura 21: Menú-Opción Editar

El sistema nos permite introducir un nuevo proyecto, una nueva versión, un nuevo CSCI, un nuevo HWCI, nuevos componentes o incidencias. La ventana que aparece en la aplicación a la hora de introducir una versión es la siguiente:

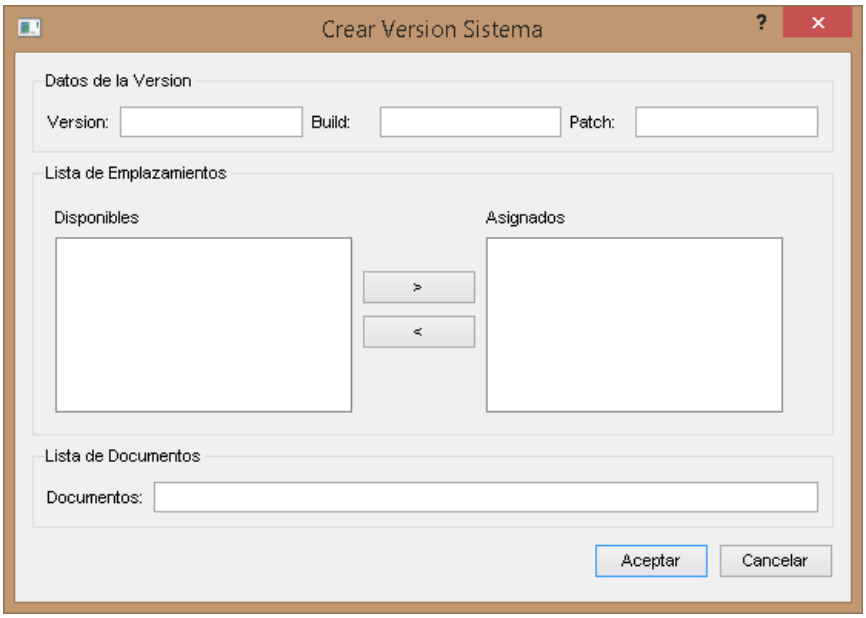


Figura 22: Ventana de nueva versión

Como se puede observar, el sistema nos permite introducir toda la información relativa a una versión, además de poder asociar dicha versión a los emplazamientos que deseemos.

El resto de acciones que se pueden realizar en el menú Editar, muestran ventanas similares a la anterior y permiten realizar las acciones descritas previamente.

La otra acción que el sistema permite realizar al usuario en el menú superior es la de importar los ficheros csv con toda la información correspondiente a una versión.

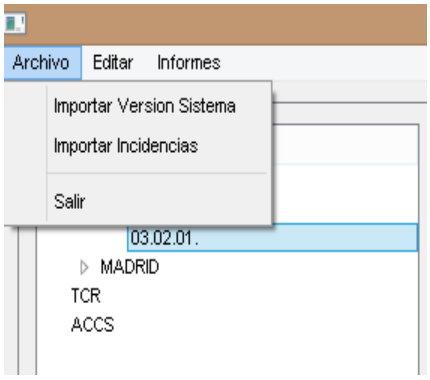


Figura 23: Menú-Opción Archivo

Se deben distinguir dos etapas para poder guardar en el sistema toda la información relativa a una versión:

- **Importación de CSCIs**. Los desarrolladores, cuando tienen una versión del sistema preparada, generan unos ficheros csv con toda la información necesaria. Se genera un fichero csv por cada CSCI, y en cada uno de ellos se

encuentra la información sobre los componentes, con sus respectivas versiones, checksums e incidencias asociadas.

```
#;-------------------------------------------------------------------------------------------;;;;;
#; Informe de control de configuracion                                                      ;;;;;
#;COMETA;;;;;
#;03.02.01;;;;;
#;CWP;;;;;
#;-------------------------------------------------------------------------------------------;;;;;
# CSC       ;CSC al que pertenece el componente                                             ;;;;;
# NOMBRE    ;nombre completo del componente, sin version                                    ;;;;;
# VERSION   ;version asignada al componente                                                 ;;;;;
# CHECKSUM  ;codigo de comprobacion del componente (sum -r)                                 ;;;;;
# SPTR      ;listado de SPTRs y UCRs que implementa el componente, separado por comas       ;;;;;
# ACCION    ;indica si el componente es nuevo (CRE), modificado (MOD) o eliminado (ELI)     ;;;;;
# NOTAS     ;notas de entrega opcionales                                                    ;;;;;
#;-------------------------------------------------------------------------------------------;;;;;
COMETA;03.02.01;CWP;
# CSC;NOMBRE;VERSION;CHECKSUM;SPTR;ACCION;NOTAS
CWPA;CWP_CWPA_bin_rh132_SDCXXIRUN;02.01;25769;SCV#2345,SCV#6765;CRE;
CWPA;CWP_CWPA_cfg_multi_asound.conf;02.01;26941;;CRE;
CWPA;CWP_CWPA_cfg_multi_conftraz.ini;02.01;52606;;CRE;
CWPA;CWP_CWPA_cfg_multi_idespmad.ini;02.01;49263;;CRE;
CWPA;CWP_CWPA_cfg_multi_idiomale.ini;02.01;54797;;CRE;
CWPA;CWP_CWPA_cfg_multi_idiomesp.ini;02.01;46766;;CRE;
CWPA;CWP_CWPA_cfg_multi_idiomfra.ini;02.01;32031;;CRE;
CWPA;CWP_CWPA_cfg_multi_idioming.ini;02.01;34742;;CRE;
CWPA;CWP_CWPA_cfg_multi_idiomita.ini;02.01;43168;;CRE;
CWPA;CWP_CWPA_cfg_multi_parampue.txt;02.01;62882;;CRE;
CWPA;CWP_CWPA_cfg_multi_versions.txt;02.01;46620;;CRE;
CWPA;CWP_CWPA_img_rh132_altavoz1.bmp;02.01;63985;;CRE;
CWPA;CWP_CWPA_img_rh132_Altavoz.bmp;02.01;2535;;CRE;
CWPA;CWP_CWPA_img_rh132_arriba.bmp;02.01;34671;;CRE;
CWPA;CWP_CWPA_img_rh132_bitm1000.bmp;02.01;12413;;CRE;
CWPA;CWP_CWPA_img_rh132_bitmap1.bmp;02.01;39229;;CRE;
```

Figura 24: Fichero CWP

- **Importación incidencias**. Simultáneamente, el equipo de desarrollo genera otro fichero específico para incidencias, con el cual se puede obtener toda la información de la misma.

```
#;-------------------------------------------------------------------------------------------;;;;;
#; Informe de control de configuracion                                                      ;;;;;
#;COMETA;;;;;
#;03.02.01;;;;;
#;-------------------------------------------------------------------------------------------;;;;;
# IDINCIDENCIA  ;id de la incidencia                                                        ;;;;;
# SYNOPSIS      ;título de la incidencia                                                    ;;;;;
# DESCRIPTION   ;descripción de la incidencia                                               ;;;;;
# TIPO          ;tipo de incidencia que es                                                  ;;;;;
# ESTADO        ;estado en el que se encuentra la incidencia (concluded o in acceptance);;;;;
# BBDD          ;base de datos a la que pertenence                                          ;;;;;
# NOTAS    ;notas de entrega opcionales                                                     ;;;;;
#;-------------------------------------------------------------------------------------------;;;;;
#COMETA;03.02.01;
#IDINCIDENCIA;SYNOPSIS;DESCRIPTION;TIPO;ESTADO;BBDD;NOTAS
SCV#11;Caída de la posición;Al realizar una llamada por acceso instantaneo se produce una caída de la posición.;SPTR;Concluded;SCV;
SCV#2345;Pérdida de audio en conferencia;Cuando tenemos una conferencia con 5 o más conferenciantes, el audio no se escucha en todas las posiciones.;SPTR;C
SCV#6765;Señalización llamadas por R2;No se señalizan en la posición las llamadas entrantes por R2;SPTR;Concluded;SCV;
```

Figura 25: Fichero Incidencias

En el sistema, para poder importar estos ficheros aparecen las siguientes ventanas:

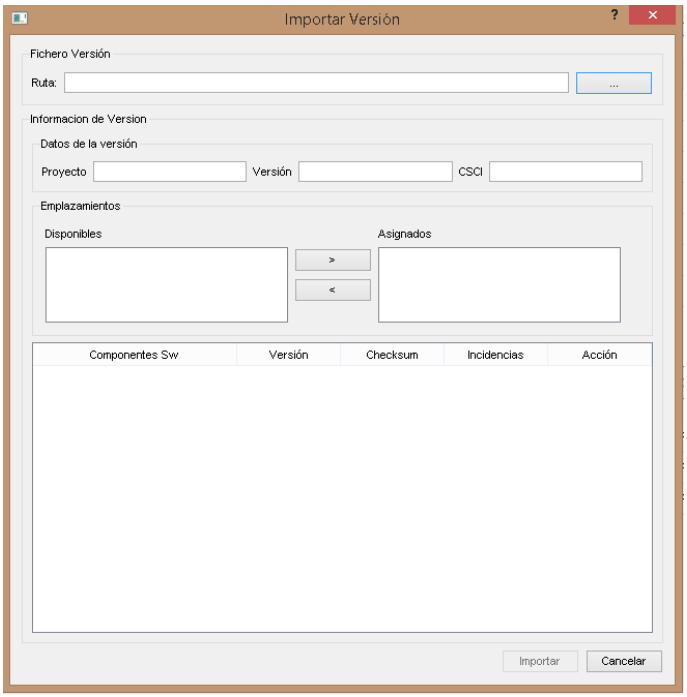


Figura 26: Ventana de importar Versión

El usuario busca el fichero del CSCI que desea importar. Cuando selecciona el fichero, automáticamente se mostrará en pantalla la información que contiene el mismo, con el objetivo de corrección de errores de manera visual. También nos mostrará la lista de emplazamientos y aquellos en los cuales ya se encuentra esa versión instalada (y almacenada en el sistema).

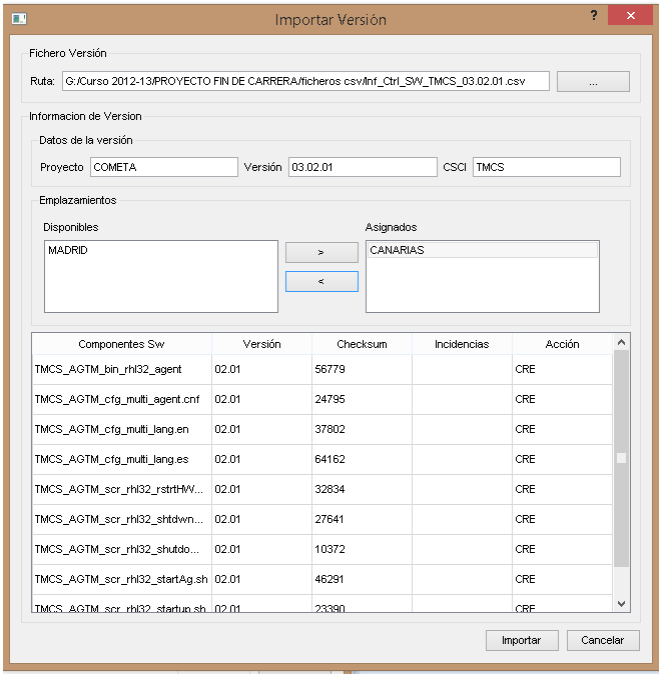


Figura 27: Carga de fichero

Una vez inspeccionado el fichero visualmente, el usuario puede definitivamente importar el fichero en el sistema y almacenar la información en la base de datos. La aplicación informa de la acción.

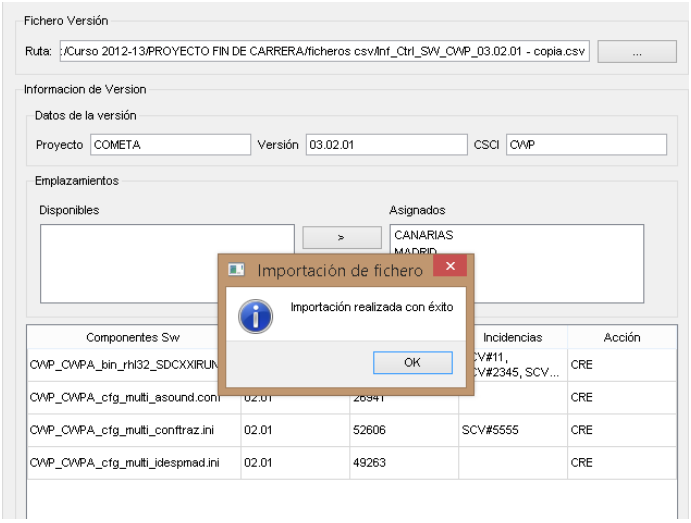


Figura 28: Importación realizada

Con respecto a las incidencias, la ventana es más simple. No se muestra previamente la información que contiene el fichero. Simplemente, se le pide al usuario que seleccione el fichero correspondiente y se realiza la importación. No se muestra la información que contiene el fichero previamente, ya que no es necesaria la comprobación previa del fichero, ya que dicha comprobación es realizada cuando el usuario importa los ficheros de los CSCIs.

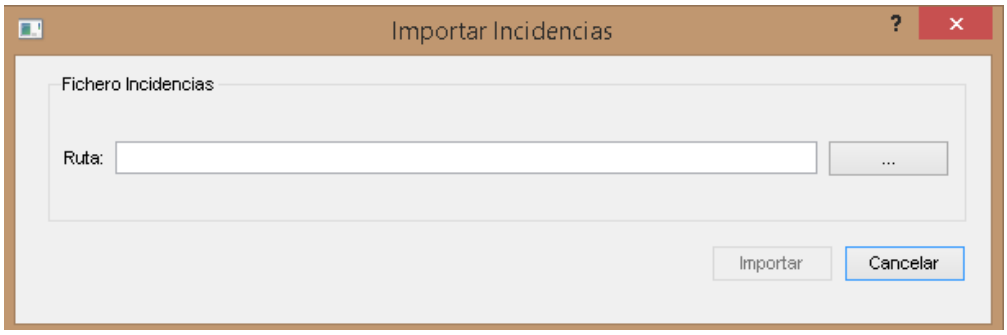


Figura 29: Ventana de importación de incidencias

La implementación del HMI se ha llevado a cabo en las siguientes clases:

- **C_Login**. Clase encargada del login inicial.
- **C_HMI**. Clase encargada en la ventana principal de la aplicación.
- **C_ProyectoDialog**. Clase encargada de la ventana mostrada para introducir información aislada de un proyecto.

- **C_VersionesSistemaDialog**. Clase encargada de la ventana mostrada para introducir información aislada de una versión.
- **C_PtrDialog**. Clase encargada de la ventana mostrada para introducir información aislada de una incidencia.
- **C_ComponentesSwDialog**. Clase encargada de la ventana mostrada para introducir información aislada de un componente.
- **C_CsciDialog**. Clase encargada de la ventana mostrada para introducir información aislada de un CSCI.
- **C_HwciDialog**. Clase encargada de la ventana mostrada para introducir información aislada de un HWCI.
- **C_ImportarDialog**. Clase encargada de la ventana mostrada para importar un fichero csv con la información de los CSCIs.
- **C_ImportarIncidenciaDialog**. Clase encargada de la ventana mostrada para importar un fichero csv con la información de las incidencias.
- **C_TabbedFileReader**. Clase que contiene los métodos necesarios para leer los ficheros csv.
- **C_ListSelectionWidget**. Clase que contiene los métodos necesarios para mostrar el widget de selección de elementos de una lista (lista de emplazamientos).

5. PLAN DE PRUEBAS

Forman parte del proceso de control de calidad y tienen como objetivo proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada.

En el presente capítulo se presentarán el conjunto de pruebas a realizar para verificar los requisitos establecidos. Todas las pruebas tienen que ser superadas para dar por concluido con éxito el desarrollo del proyecto. Además, se mostrará una matriz que relaciona el conjunto de pruebas con los requisitos software.

5.1 PRUEBAS DE ACEPTACIÓN

Una vez terminado el diseño e implementación de la aplicación, se llevaron a cabo una serie de pruebas de aceptación, con el objetivo de determinar la calidad del producto desarrollado. Estas pruebas fueron realizadas en dos fases: una **primera fase** interna, previa a la entrega a cliente, cuya finalidad era determinar errores que se pudieran corregir y evitar que aparecieran en la **segunda fase**, fase que se llevó a cabo en presencia del cliente y donde se podían verificar los requisitos que se establecieron inicialmente para el diseño de la aplicación.

Las pruebas de aceptación del sistema se definen de la siguiente forma:

- Identificador: está compuesto por las iniciales "PA" seguidas del número de la prueba.
- Entrada esperada: acción que debe realizarse para llevar a cabo la prueba.
- Salida esperada: resultado que debe obtenerse para que la prueba se valide con éxito.

PA-01	
Entrada esperada	El usuario abre la aplicación.
Salida esperada	Se muestra la ventana de Login.

Tabla 31: PA-01 Inicio aplicación

PA-02	
Entrada esperada	El usuario se loguea incorrectamente en la aplicación.
Salida esperada	El sistema no permite al usuario entrar en la aplicación pidiéndole de nuevo la contraseña.

Tabla 32: PA-02 Login

PA-03	
Entrada esperada	El usuario se loguea correctamente en la aplicación.
Salida esperada	El sistema muestra la ventana principal de la aplicación.

Tabla 33: PA-03 Login

PA-04	
Entrada esperada	El usuario selecciona un proyecto que se encuentre en el árbol de proyectos.
Salida esperada	Si dicho proyecto tiene emplazamientos asociados, se despliegan a continuación del proyecto.

Tabla 34: PA-04 Árbol de proyectos

PA-05	
Entrada esperada	El usuario selecciona un emplazamiento que tenga asociado el proyecto.
Salida esperada	Si dicho emplazamiento tiene versiones del sistema asociadas, se despliegan a continuación del emplazamiento.

Tabla 35: PA-05 Árbol de proyectos

PA-06	
Entrada esperada	El usuario selecciona una versión del sistema que tenga asociada el emplazamiento.
Salida esperada	Se muestra la información relativa a dicha versión en la tabla de componentes.

Tabla 36: PA-06 Tabla de componentes

PA-07	
Entrada esperada	El usuario selecciona un componente de la tabla de componentes.
Salida esperada	Si dicho componente tiene incidencias asociadas, se muestran en la tabla de incidencias.

Tabla 37: PA-07 Tabla de incidencias

PA-08	
Entrada esperada	El usuario selecciona la opción “Importar versión sistema” del menú “Archivo”.
Salida esperada	Se abre una nueva ventana que permite al usuario importar un fichero csv.

Tabla 38: PA-08 Importar Versión Sistema

PA-09	
Entrada esperada	El usuario selecciona el fichero csv que desea importar.
Salida esperada	Se muestra en pantalla la información que contiene el fichero y se habilita el botón importar.

Tabla 39: PA-09 Importar Versión Sistema

PA-10	
Entrada esperada	El usuario pulsa cancelar.
Salida esperada	La ventana de importación se cierra y no se almacena la información en el sistema.

Tabla 40: PA-10 Importar Versión Sistema

PA-11	
Entrada esperada	El usuario pulsa importar.
Salida esperada	La aplicación muestra un mensaje de confirmación y posteriormente un mensaje para informar de que la importación ha sido realizada con éxito.

Tabla 41: PA-11 Importar Versión Sistema

PA-12	
Entrada esperada	El usuario selecciona la opción “Importar incidencias” del menú “Archivo”.
Salida esperada	Se abre una nueva ventana que permite al usuario importar un fichero csv.

Tabla 42: PA-12 Importar Incidencias

PA-13	
Entrada esperada	El usuario selecciona el fichero csv que desea importar.
Salida esperada	Se habilita el botón importar.

Tabla 43: PA-13 Importar Incidencias

PA-14	
Entrada esperada	El usuario pulsa cancelar.
Salida esperada	La ventana de importación se cierra y no se almacena la información en el sistema.

Tabla 44: PA-14 Importar Incidencias

PA-15	
Entrada esperada	El usuario pulsa importar.
Salida esperada	La aplicación muestra un mensaje de confirmación y posteriormente un mensaje para informar de que la importación ha sido realizada con éxito.

Tabla 45: PA-15 Importar Incidencias

PA-16	
Entrada esperada	El usuario selecciona la opción “Salir” del menú “Archivo”.
Salida esperada	El usuario sale de la aplicación. La aplicación se cierra.

Tabla 46: PA-16 Salir de la aplicación

PA-17	
Entrada esperada	El usuario selecciona cualquier opción del menú “Editar”.
Salida esperada	La aplicación muestra una ventana en relación a la opción seleccionada y permite al usuario introducir información aislada al sistema.

Tabla 47: PA-17 Opción Editar

A continuación se muestra la matriz que relaciona el conjunto de pruebas software con los requisitos que se definieron para el desarrollo de la aplicación.

	PA-01	PA-02	PA-03	PA-04	PA-05	PA-06	PA-07	PA-08	PA-09	PA-10	PA-11	PA-12	PA-13	PA-14	PA-15	PA-16	PA-17
RSF-01	X																
RSF-02	X																
RSF-03		X															
RSF-04		X															
RSF-05			X	X													
RSF-06								X				X					
RSF-07								X				X					
RSF-08																	X
RSF-09								X									
RSF-10												X					
RSF-11																X	
RSF-12																	X
RSF-13									X								
RSF-14									X		X						
RSF-15										X				X			
RSF-16									X								
RSF-17															X		
RSF-18													X				
RSF-19				X													
RSF-20					X												
RSF-21						X											
RSF-22							X										

Tabla 48: Matriz RSF x PA

5.2 RESULTADO DE LAS PRUEBAS

Una vez finalizado el desarrollo de la aplicación y pasado el conjunto de pruebas software, se puede observar que, se han conseguido unos resultados satisfactorios y se han ejecutado correctamente todas las pruebas mencionadas en el apartado anterior.

6. GESTIÓN DEL PROYECTO

En el siguiente capítulo, se expondrá la gestión que ha tenido el proyecto de manera detallada. En primer lugar, se explicarán las fases que han sido planificadas y por las que ha pasado el proyecto a lo largo de su desarrollo. En segundo y último lugar, se realizará un análisis del coste que ha supuesto el desarrollo de la aplicación.

6.1 PLANIFICACIÓN DEL PROYECTO

Las etapas que se han seguido durante todo el desarrollo del proyecto son las mencionadas anteriormente en el apartado 4.1.1. De manera resumida, estas etapas son las siguientes:

- **Fase de especulación**.
  - Definición de requisitos.
  - Características software del sistema.
  - Planificación.

- **Fase de colaboración**.
  - Descripción / Diseño del sistema.
  - Implementación.

- **Fase de aprendizaje**.
  - Rendimiento y calidad.
  - Mejoras y errores.
  - Conclusiones.

El tiempo dedicado a cada una de las etapas queda reflejado en el siguiente diagrama de Gantt:

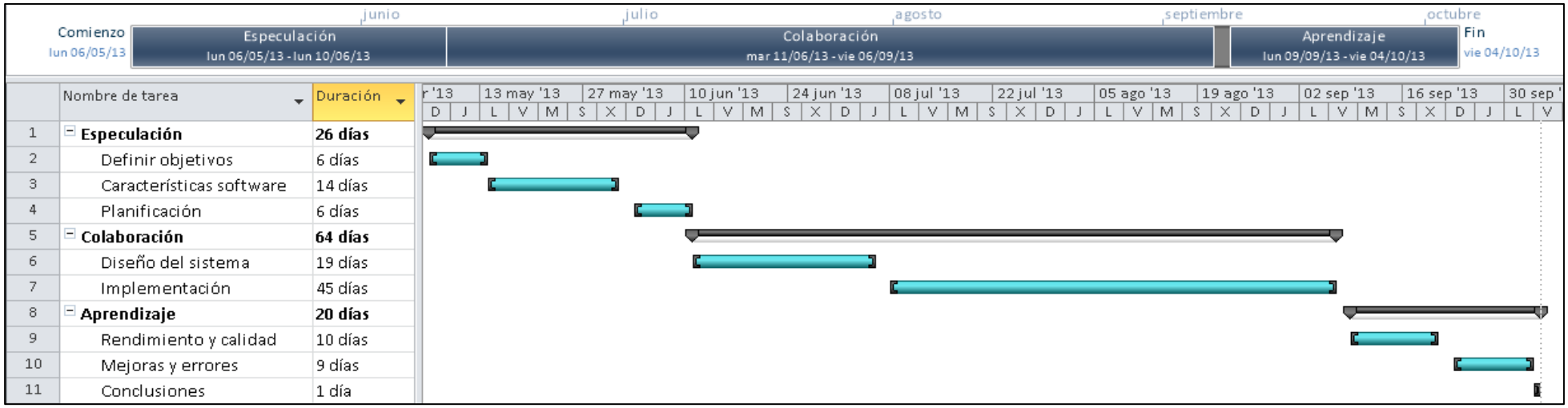


Figura 30: Planificación del proyecto

6.2 PRESUPUESTO

Para determinar el coste total de la aplicación, se debe diferenciar entre los costes debido al personal que interviene en el proyecto, los costes debido al material hardware empleado, los costes debido al material software empleado y los costes indirectos como pueden ser la luz, la conexión a internet, etc.

6.2.1 Coste personal

Dentro de estos costes, se han incluido al desarrollador de la aplicación como a los asesores que han ayudado al desarrollador a llevar a cabo dicho proyecto. En la siguiente tabla se detallan estos costes:

Personal	Categoría	Coste Hombre/Hora	Dedicación (horas)	Coste (euros)
Roldán Tormo, Rafael	Ingeniero Junior	20	600	12.000
Serna Humanes, Hugo	Ingeniero Senior	32	40	1.280
Fernández Martínez, José Miguel	Ingeniero Senior	32	40	1.280
Salgado Lorenzo, Yajaira	Asesor	80	10	800
Almenares Mendoza, Florina	Asesor	80	10	800
			Total	16.160

Tabla 49: Coste de personal

6.2.2 Coste hardware

En este apartado, se tendrán en cuenta los costes asociados al material hardware empleado. Para ello se debe tener en cuenta las amortizaciones de este material, prefijando la vida útil y el tiempo que se ha utilizado el material. En la siguiente tabla se detallan estos costes:

Equipo	Coste (euros)	Vida útil estimada (meses)	Tiempo de uso (meses)	Coste para el proyecto (euros)
Ordenador Sony Vaio, Serie VGN-NS11S	950	60	3	47.5
			Total	47.5

Tabla 50: Coste hardware

6.2.3 Coste software

En este apartado, se tendrán en cuenta los costes asociados al material software empleado.

Material software	Coste (euros)	Coste para el proyecto (euros)
Microsoft Office hogar y estudiantes 2010	90	90
	Total	90

Tabla 51: Coste software

6.2.4 Costes indirectos

En este apartado, se incluyen costes que no pueden contabilizarse de manera exacta, como son el uso de internet, la luz empleada, la licencia de Microsoft Visual Studio que se utiliza en todo el departamento de SCV de INDRA, etc.

Se ha decidid aplicar un 20% de los costes totales, y queda recogido en la siguiente tabla:

Tipo	Porcentaje (%)	Coste (euros)
Costes totales		16.297,5
Costes indirectos	20	3.259,5

Tabla 52: Costes indirectos

6.2.5 Costes totales

Por último, se muestra el coste total del proyecto a continuación:

Presupuesto	Coste (euros)
Coste personal	16.160
Coste hardware	47.5
Coste software	90
Coste indirecto	3.259,5
Total	19.557

Tabla 53: Coste total

En resumen, el coste total del proyecto desarrollado es de **19.557 euros.**

7. CONCLUSIONES Y LÍNEAS FUTURAS

Tras la realización del proyecto, por último, en el siguiente capítulo se expondrán las conclusiones obtenidas tras el desarrollo. Además, se enumerarán las posibles líneas futuras, con el objetivo de mejorar la aplicación en una segunda iteración de la metodología ágil ASD, que se recuerda que es la metodología empleada en el proyecto.

7.1 CONCLUSIONES

La implementación de un sistema para gestión de versiones software para el departamento SCV (Sistemas de Comunicaciones de Voz sobre IP) ha sido un proyecto que se presentó como un gran reto en sus inicios. Un gran reto debido a la dificultad que suponía el desarrollo de una aplicación software sin tener apenas conocimientos sobre la materia (bases de datos y programación en C++), al que se le sumaba la necesidad de realizar un buen trabajo para que el departamento SCV pudiera utilizar la aplicación en los plazos acordados.

El desconocimiento sobre las bases de datos y sobre la programación en C++ por otro lado era un factor positivo, debido a que ayudaría a ampliar conocimientos sobre temas que no se han tocado a lo largo de la carrera universitaria.

El desarrollo de una aplicación para una empresa como INDRA SISTEMAS, hacía también que el proyecto fin de carrera fuera más atractivo, ya que el trabajo realizado tendría sus frutos y sería usado por una empresa española de gran nombre en el mercado internacional. Además, la realización del proyecto en INDRA, supone una primera toma de contacto del alumno con el mundo laboral y sirve para tener una idea de cuál sería una salida laboral en un futuro.

Finalmente, se ha conseguido cumplir con los objetivos propuestos en un principio para el desarrollo del sistema, dando lugar a una aplicación que facilita de manera considerable el trabajo que realizaba hasta ahora el departamento SCV. Gracias a la aplicación, se ha sustituido el fichero Excel que hasta entonces utilizaban, por la gestión del versionado gracias a la aplicación desarrollada, aumentado la eficacia, la eficiencia y seguridad en la información tratada.

Cuando hay una versión lista para ser lanzada en alguno de los aeropuertos en los que trabaja el departamento SCV, el equipo de desarrollo envía los ficheros csv con toda la información relativa a la versión. Esta información es importada a la aplicación desarrollada, por lo que a partir de ese momento, la información está accesible de manera rápida y fácil para cualquier empleado del departamento, sin necesidad de tener un Excel, donde hay que buscar entre los miles de registros que puede tener el fichero.

La aplicación se sigue desarrollando en la actualidad, incluyendo mejoras y funcionalidades que quedan fuera de los objetivos del proyecto fin de carrera. Estas mejoras y funcionalidades fueron detectadas durante el control de calidad y se incluyen dentro de las líneas futuras que puede tener el proyecto fin de carrera.

7.2 LINEAS FUTURAS

Como ya se ha mencionado, se encontraron una serie de mejoras durante el control de calidad que hacen que la aplicación siga actualmente en desarrollo, con una segunda iteración en su ciclo de vida, con el objetivo de introducir las mejoras y las nuevas funcionalidades.

7.2.1 Generación de informes

Teniendo toda la información sobre una versión en el sistema, surgió la idea de crear informes a partir de esa información. Estos informes serían de gran ayuda para los instaladores, ya que, imprimiendo estos informes, tendrían a mano la información sobre la versión a instalar en el aeropuerto. También, estos informes seguirán un modelo aprobado por el departamento de calidad de INDRA, con el objetivo de que sean entregados a los clientes junto con la versión lanzada.

7.2.2 Conexión con IBM Rational Change

El departamento SCV tiene una base de datos para la gestión de incidencias. Esta base de datos, IBM Rational Change, se utiliza para el control y gestión de todas las incidencias que hay en los sistemas de comunicaciones de voz.

La idea es la de conectar la aplicación desarrollada a la base de datos de Change, con el objetivo de evitar tener que importar un fichero csv a la aplicación y directamente gestionar las incidencias gracias a la conexión a la base de datos de Change.

7.2.3 Aplicación Cliente-Servidor

Actualmente, la aplicación se encuentra instalada en un ordenador portátil, y es en ese ordenador donde se realizan todas las gestiones sobre el versionado.

El objetivo es la de adaptar la aplicación para tener un servidor y así poder tener numerosos clientes y no depender de un único ordenador portátil.

7.2.4 Roles

La aplicación no distingue entre distintos roles de usuario. Cualquier usuario que utilice la aplicación actualmente (logueandose previamente con una contraseña genérica), puede realizar las mismas acciones.

El objetivo es la de diferenciar entre distintos grupos de usuario en función del registro realizado en la aplicación (administrador, usuario regular, usuario invitado, etc.).

BIBLIOGRAFÍA

[1] Elmasri, Ramez and Navathe, Shamkant. “Database systems: models, languages, design, and application programming”, 6th Edition. Pearson Education. 10/10/2013.

[2] Mª Victoria Nevado Cabello. “Introducción a las bases de datos relacionales”. Editorial Vision Libros. 10/10/2013.

[3] Bruce Momjian. “PostgreSQL: introduction and concepts”. Addison-Wesley. 12/10/2013.

[4] PostgreSQL (http://www.postgresql.org). 03/01/2014.

[5] Matthias Kalle Dalheimer. “Programming with Qt: Writing Portable GUI applications on Unix and Win32”, 2nd Edition. O’Reilly. 05/11/2013.

[6] Jasmin Blanchette, Mark Summerfield. “C++ GUI Programming with Qt 4”, 2nd Edition. Prentice Hall. 05/11/2013.

[7] QT (http://qt.digia.com). 03/01/2014.

[8] Jeff Ferguson, Brian Patterson, Jason Beres. “La biblia de C++”. Editorial Anaya. 02/11/2013.

[9] Microsoft SQL Server (http://www.microsoft.com/es-es/sqlserver). 15/11/2013.

[10] Oracle (http://www.oracle.com/us/products/database). 15/11/2013.

[11] MySQL (http://www.mysql.com). 15/11/2013.

[12] Allen B. Tucker. “Programming Languages”.Mc-Graw-Hill Edition. 02/11/2013.

[13] Felix Garcia Carballeira. “El lenguaje de programación C: diseño e implementación de programas”. Prentice Hall. 02/11/2013.

[14] Eckel, B. “Piensa en Java”. Prentice Hall. 02/11/2013.

[15] Jennifer Vesperman. “Essential CVS”, 2nd Edition. O’Reilly. 13/12/2013.

[16] C. Michael Pilato, Ben Collins-Sussman, Brian W. Fitzpatrick. “Version Control with Subversion”, 2nd Edition. O’Reilly. 13/12/2013.

[17] Jon Loeliger, Matthew McCullough. “Version Control with Git”, 2nd Edition. O’Reilly. 13/12/2013.

[18] Metodologías ágiles en el desarrollo de software (http://noqualityinside.com.ar/nqi/nqifiles/XP_Agil.pdf). 02/01/2014.

[19] James A. Highsmith. “Adaptative software development: A collaborative approach to managing complex systems”. Dorset House Publishing. 20/11/2013.